

# OPERATOR'S MANUAL



## **CR200/CR200X Series** **Dataloggers**

Revision: 3/15



Copyright © 2000-2015  
Campbell Scientific, Inc.



# Limited Warranty

---

The CR200X is warranted for thirty-six (36) months (CR200-series dataloggers warranted for twelve (12) months) subject to this limited warranty:

“Products manufactured by CSI are warranted by CSI to be free from defects in materials and workmanship under normal use and service for twelve months from the date of shipment unless otherwise specified in the corresponding product manual. (Product manuals are available for review online at [www.campbellsci.com](http://www.campbellsci.com).) Products not manufactured by CSI, but that are resold by CSI, are warranted only to the limits extended by the original manufacturer. Batteries, fine-wire thermocouples, desiccant, and other consumables have no warranty. CSI’s obligation under this warranty is limited to repairing or replacing (at CSI’s option) defective Products, which shall be the sole and exclusive remedy under this warranty. The Customer assumes all costs of removing, reinstalling, and shipping defective Products to CSI. CSI will return such Products by surface carrier prepaid within the continental United States of America. To all other locations, CSI will return such Products best way CIP (port of entry) per Incoterms ® 2010. This warranty shall not apply to any Products which have been subjected to modification, misuse, neglect, improper service, accidents of nature, or shipping damage. This warranty is in lieu of all other warranties, expressed or implied. The warranty for installation services performed by CSI such as programming to customer specifications, electrical connections to Products manufactured by CSI, and Product specific training, is part of CSI’s product warranty. **CSI EXPRESSLY DISCLAIMS AND EXCLUDES ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. CSI hereby disclaims, to the fullest extent allowed by applicable law, any and all warranties and conditions with respect to the Products, whether express, implied or statutory, other than those expressly provided herein.”**

# Assistance

---

Products may not be returned without prior authorization. The following contact information is for US and international customers residing in countries served by Campbell Scientific, Inc. directly. Affiliate companies handle repairs for customers within their territories. Please visit [www.campbellsci.com](http://www.campbellsci.com) to determine which Campbell Scientific company serves your country.

To obtain a Returned Materials Authorization (RMA), contact CAMPBELL SCIENTIFIC, INC., phone (435) 227-9000. After an application engineer determines the nature of the problem, an RMA number will be issued. Please write this number clearly on the outside of the shipping container. Campbell Scientific's shipping address is:

**CAMPBELL SCIENTIFIC, INC.**

RMA# \_\_\_\_\_  
815 West 1800 North  
Logan, Utah 84321-1784

For all returns, the customer must fill out a "Statement of Product Cleanliness and Decontamination" form and comply with the requirements specified in it. The form is available from our web site at [www.campbellsci.com/repair](http://www.campbellsci.com/repair). A completed form must be either emailed to [repair@campbellsci.com](mailto:repair@campbellsci.com) or faxed to (435) 227-9106. Campbell Scientific is unable to process any returns until we receive this form. If the form is not received within three days of product receipt or is incomplete, the product will be returned to the customer at the customer's expense. Campbell Scientific reserves the right to refuse service on products that were exposed to contaminants that may cause health or safety concerns for our employees.

# Precautions

---

**DANGER — MANY HAZARDS ARE ASSOCIATED WITH INSTALLING, USING, MAINTAINING, AND WORKING ON OR AROUND TRIPODS, TOWERS, AND ANY ATTACHMENTS TO TRIPODS AND TOWERS SUCH AS SENSORS, CROSSARMS, ENCLOSURES, ANTENNAS, ETC.** FAILURE TO PROPERLY AND COMPLETELY ASSEMBLE, INSTALL, OPERATE, USE, AND MAINTAIN TRIPODS, TOWERS, AND ATTACHMENTS, AND FAILURE TO HEED WARNINGS, INCREASES THE RISK OF DEATH, ACCIDENT, SERIOUS INJURY, PROPERTY DAMAGE, AND PRODUCT FAILURE. TAKE ALL REASONABLE PRECAUTIONS TO AVOID THESE HAZARDS. CHECK WITH YOUR ORGANIZATION'S SAFETY COORDINATOR (OR POLICY) FOR PROCEDURES AND REQUIRED PROTECTIVE EQUIPMENT PRIOR TO PERFORMING ANY WORK.

Use tripods, towers, and attachments to tripods and towers only for purposes for which they are designed. Do not exceed design limits. Be familiar and comply with all instructions provided in product manuals. Manuals are available at [www.campbellsci.com](http://www.campbellsci.com) or by telephoning (435) 227-9000 (USA). You are responsible for conformance with governing codes and regulations, including safety regulations, and the integrity and location of structures or land to which towers, tripods, and any attachments are attached. Installation sites should be evaluated and approved by a qualified engineer. If questions or concerns arise regarding installation, use, or maintenance of tripods, towers, attachments, or electrical connections, consult with a licensed and qualified engineer or electrician.

## General

- Prior to performing site or installation work, obtain required approvals and permits. Comply with all governing structure-height regulations, such as those of the FAA in the USA.
- Use only qualified personnel for installation, use, and maintenance of tripods and towers, and any attachments to tripods and towers. The use of licensed and qualified contractors is highly recommended.
- Read all applicable instructions carefully and understand procedures thoroughly before beginning work.
- Wear a **hardhat** and **eye protection**, and take **other appropriate safety precautions** while working on or around tripods and towers.
- **Do not climb** tripods or towers at any time, and prohibit climbing by other persons. Take reasonable precautions to secure tripod and tower sites from trespassers.
- Use only manufacturer recommended parts, materials, and tools.

## Utility and Electrical

- **You can be killed** or sustain serious bodily injury if the tripod, tower, or attachments you are installing, constructing, using, or maintaining, or a tool, stake, or anchor, come in **contact with overhead or underground utility lines**.
- Maintain a distance of at least one-and-one-half times structure height, 20 feet, or the distance required by applicable law, **whichever is greater**, between overhead utility lines and the structure (tripod, tower, attachments, or tools).
- Prior to performing site or installation work, inform all utility companies and have all underground utilities marked.
- Comply with all electrical codes. Electrical equipment and related grounding devices should be installed by a licensed and qualified electrician.

## Elevated Work and Weather

- Exercise extreme caution when performing elevated work.
- Use appropriate equipment and safety practices.
- During installation and maintenance, keep tower and tripod sites clear of un-trained or non-essential personnel. Take precautions to prevent elevated tools and objects from dropping.
- Do not perform any work in inclement weather, including wind, rain, snow, lightning, etc.

## Maintenance

- Periodically (at least yearly) check for wear and damage, including corrosion, stress cracks, frayed cables, loose cable clamps, cable tightness, etc. and take necessary corrective actions.
- Periodically (at least yearly) check electrical ground connections.

WHILE EVERY ATTEMPT IS MADE TO EMBODY THE HIGHEST DEGREE OF SAFETY IN ALL CAMPBELL SCIENTIFIC PRODUCTS, THE CUSTOMER ASSUMES ALL RISK FROM ANY INJURY RESULTING FROM IMPROPER INSTALLATION, USE, OR MAINTENANCE OF TRIPODS, TOWERS, OR ATTACHMENTS TO TRIPODS AND TOWERS SUCH AS SENSORS, CROSSARMS, ENCLOSURES, ANTENNAS, ETC.



# Table of Contents

---

<b>Section 1. Introduction .....</b>	<b>1</b>
1.1 CR200(X) series Datalogger Models .....	1
<b>Section 2. Quickstart Tutorial .....</b>	<b>3</b>
2.1 Primer - CR200(X) Data Acquisition .....	3
2.1.1 Components of a Data Acquisition System .....	3
2.1.2 CR200(X) Mounting .....	4
2.1.3 Wiring Panel .....	4
2.1.4 Battery Backup .....	5
2.1.5 Power Supply .....	5
2.1.6 Antenna .....	5
2.1.7 Analog Sensors .....	6
2.1.8 Bridge Sensors .....	6
2.1.9 Pulse Sensors .....	7
2.1.10 Digital I/O Ports .....	8
2.1.11 RS-232 Sensors .....	9
2.2 Hands-on Exercise - Measuring Temperature .....	10
2.2.1 Hardware Setup .....	11
2.2.2 Configuration .....	11
2.2.3 PC200W Software Setup .....	11
<b>Section 3. Overview .....</b>	<b>23</b>
3.1 CR200(X) Overview .....	23
3.1.1 Programmed Instructions Are Evaluated Sequentially .....	24
3.1.2 Sensor Support .....	25
3.1.3 Input / Output Interface: The Wiring Panel .....	25
3.1.4 Power Requirements .....	29
3.1.5 Programming: Firmware and User Programs .....	29
3.1.6 Memory and Data Storage .....	30
3.1.7 Communications Overview .....	31
3.1.8 Maintenance Overview .....	32
3.2 PC Support Software .....	34
3.3 Specifications .....	35
<b>Section 4. Sensor Support .....</b>	<b>37</b>
4.1 Powering Sensors .....	37
4.1.1 Switched Precision .....	37
4.1.2 Continuous Unregulated (Nominal 12 Volt) .....	37
4.1.3 Switched Unregulated (Nominal 12 Volt) .....	38
4.2 Voltage Measurement .....	38
4.2.1 Measurement Sequence .....	38
4.2.2 Measurement Accuracy .....	38
4.2.3 Voltage Range .....	40
4.2.4 Integration .....	40
4.2.5 Self-Calibration .....	41
4.3 Bridge Resistance Measurements .....	41
4.3.1 Measurements Requiring AC Excitation .....	42

4.4	Pulse Count Measurement.....	42
4.4.1	Pulse input Channels .....	43
4.4.2	Pulse Input on Digital I/O Channels C1–C2 .....	45
4.5	Period Averaging Measurements .....	45
4.6	SDI-12 Recording .....	46
4.7	Cabling Effects on Measurements.....	46
4.7.1	Analog Sensor Cables .....	46
4.7.2	Pulse Sensors.....	46
4.7.3	Serial Sensors.....	47
<b>Section 5. Measurement and Control Peripherals..</b>		<b>49</b>
5.1	Control Output .....	49
5.1.1	Binary Control.....	49
5.2	Other Peripherals.....	51
5.2.1	TIMs.....	51
<b>Section 6. CR200(X) Power Supply .....</b>		<b>53</b>
6.1	Power Requirement .....	53
6.2	Calculating Power Consumption.....	53
6.3	Power Supplies.....	53
6.3.1	Battery Connection.....	53
<b>Section 7. Grounding .....</b>		<b>55</b>
7.1	ESD Protection.....	55
7.1.1	Lightning Protection.....	56
7.2	Single-Ended Measurement Reference .....	57
<b>Section 8. CR200(X) Configuration .....</b>		<b>59</b>
8.1	DevConfig.....	59
8.2	Sending the Operating System .....	60
8.2.1	Sending OS with DevConfig.....	60
8.3	Settings.....	62
8.3.1	Settings via DevConfig .....	62
8.3.2	Settings via CRBASIC.....	66
8.3.3	Settings via Terminal Emulator.....	66
8.3.4	Durable Settings.....	67
<b>Section 9. Programming .....</b>		<b>69</b>
9.1	Inserting Comments into Program.....	69
9.2	Sending Programs .....	69
9.3	Writing Programs.....	69
9.3.1	Short Cut Editor and Program Generator.....	70
9.3.2	CRBASIC Editor.....	70
9.4	Numerical Formats.....	71
9.5	Structure.....	72
9.6	Declarations I - Single-line Declarations .....	73
9.6.1	Variables .....	73
9.6.2	Constants.....	76
9.6.3	Alias and Unit Declarations .....	77



9.7	Declarations II - Declared Sequences .....	77
9.7.1	Data Tables .....	77
9.7.2	Subroutines .....	83
9.8	Program Execution Timing .....	83
9.9	Instructions.....	84
9.9.1	Measurement and Data Storage Processing .....	84
9.9.2	Parameter Types .....	85
9.9.3	Names in Parameters .....	85
9.9.4	Expressions in Parameters .....	86
9.9.5	Arrays of Multipliers and Offsets .....	86
9.10	Expressions .....	87
9.10.1	Floating Point Arithmetic .....	87
9.10.2	Mathematical Operations .....	87
9.10.3	Logical Expressions.....	88
9.11	Program Access to Data Tables.....	90

## **Section 10. CRBASIC Programming Instructions .... 93**

10.1	Program Declarations.....	93
10.1.1	Variable Declarations & Modifiers.....	93
10.1.2	Constant Declarations .....	93
10.2	Data Table Declarations .....	94
10.2.1	Data Table Modifiers .....	94
10.2.2	Data Storage Output Processing .....	94
10.3	Single Execution at Compile.....	95
10.4	Program Control Instructions .....	96
10.4.1	Common Controls.....	96
10.5	Measurement Instructions .....	98
10.5.1	Diagnostics .....	98
10.5.2	Voltage.....	98
10.5.3	Pulse.....	98
10.5.4	Digital I/O .....	99
10.5.5	SDI-12.....	99
10.6	Processing and Math Instructions.....	100
10.6.1	Mathematical Operators.....	100
10.6.2	Logical Operators .....	100
10.6.3	Trigonometric Functions.....	101
10.6.4	Arithmetic Functions .....	102
10.6.5	Spatial Processing.....	103
10.6.6	Other Functions.....	104
10.7	Clock Functions.....	104
10.8	Serial Input / Output.....	105
10.9	Peer-to-Peer PakBus Communications.....	105
10.10	Data Table Access and Management.....	106
10.11	SCADA .....	107
10.12	Satellite Systems Programming.....	108
10.12.1	GOES.....	108

## **Section 11. Programming Resource Library..... 109**

11.1	Remote Sensor Interface .....	109
11.2	Radio Power Minimization .....	110
11.3	Multiple Switch Closure Measurements .....	112
11.4	SDI-12 Sensor Support .....	112
11.4.1	SDI-12 Command Basics.....	112
11.4.2	SDI-12 Communications .....	116

11.4.3	SDI-12 Power Considerations .....	118
11.5	Wind Vector .....	120
11.5.1	OutputOpt Parameters .....	120
11.5.2	Wind Vector Processing.....	120
11.6	TrigVar and DisableVar - Controlling Data Output and Output Processing .....	125
11.7	Multiple Data Intervals in Data Tables.....	126
<b>Section 12. Memory and Data Storage .....</b>		<b>129</b>
12.1	Data Storage .....	129
12.1.1	Data Table Storage.....	129
12.2	Memory Conservation .....	130
12.3	Memory Reset .....	130
12.3.1	Full Memory Reset.....	130
12.3.2	Program Send Reset.....	130
12.3.3	Manual Data Table Reset.....	130
<b>Section 13. Telecommunications and Data Retrieval.....</b>		<b>131</b>
13.1	Hardware and Carrier Signal .....	131
13.2	Protocols.....	132
13.3	Initiating Telecommunications .....	132
13.4	Data Retrieval.....	132
13.4.1	Via Telecommunications .....	132
13.4.2	Data Format on Computer.....	132
<b>Section 14. PakBus Overview .....</b>		<b>133</b>
14.1	PakBus Addresses .....	133
14.2	Nodes: Leaf Nodes and Routers .....	133
14.3	Router and Leaf Node Configuration .....	134
14.4	Linking Nodes: Neighbor Discovery.....	134
14.4.1	Hello-message (two-way exchange) .....	135
14.4.2	Beacon (one-way broadcast) .....	135
14.4.3	Hello-request (one-way broadcast) .....	135
14.4.4	Neighbor Lists.....	135
14.4.5	Adjusting Links.....	135
14.4.6	Maintaining Links .....	136
14.5	Troubleshooting.....	136
14.5.1	Link Integrity .....	136
14.5.2	Ping .....	137
14.5.3	Traffic Flow .....	138
14.6	LoggerNet Device Map Configuration.....	138
<b>Section 15. Alternate Telecoms Resource Library ...</b>		<b>139</b>
15.1	Modbus.....	139
15.1.1	Overview.....	139
15.1.2	Terminology.....	139
15.1.3	Programming for Modbus .....	140
15.1.4	Troubleshooting .....	142

**Section 16. Support Software ..... 143**

16.1 Short Cut .....	143
16.2 PC200W .....	143
16.3 Visual Weather.....	143
16.4 PC400.....	144
16.5 RTDAQ.....	144
16.6 LoggerNet Suite .....	144
16.7 PDA Software .....	145
16.8 Network Planner.....	145

**Section 17. Care and Maintenance ..... 147**

17.1 Temperature Range .....	147
17.2 Moisture Protection.....	147
17.3 Enclosures.....	147
17.4 Replacing the Internal Battery.....	148

**Section 18. Troubleshooting ..... 151**

18.1 Programming.....	151
18.1.2 NAN and $\pm$ INF .....	153
18.2 Communications.....	154
18.2.1 RS-232 .....	154
18.2.2 Communicating with Multiple PC Programs.....	154
18.3 Power Supply .....	155
18.3.1 Overview.....	155
18.3.2 Troubleshooting at a Glance .....	155
18.3.3 Diagnosis and Fix Procedures.....	156

**Appendices****Appendix A. Glossary ..... 1**

A.1 Terms .....	1
A.2 Concepts.....	13
A.2.1 Accuracy, Precision, and Resolution .....	13

**Appendix B. Status Table and Settings ..... 15****Appendix C. Serial Port Pin Outs..... 21**

C.1 RS-232 Communications Port .....	21
C.1.1 Pin-Out.....	21

**Appendix D. ASCII / ANSI Table..... 23****Appendix E. Antenna Usage and Compliance..... 27**

E.1 Use of Antenna with CR200(X).....	27
E.2 Part 15 FCC Compliance Warning .....	27
E.2.1 Use of Approved Antennas.....	28

## Index ..... 29

### List of Figures

Figure 1: Data Acquisition System Components.....	3
Figure 2: CR200(X) Wiring Panel.....	5
Figure 3: Analog Sensor Wired to Single-Ended Channel #1 .....	6
Figure 4: Half Bridge Wiring -- Wind Vane Potentiometer .....	7
Figure 5: Pulse Input Types.....	7
Figure 6: Pulse Input Wiring -- Anemometer Switch .....	8
Figure 7: Control and Monitoring with Digital I/O .....	9
Figure 8: Location of RS-232 Port .....	10
Figure 9: Use of RS-232 when Reading RS-232 Devices .....	10
Figure 10: Power and RS-232 Connections.....	11
Figure 11: PC200W Main Window .....	13
Figure 12: Short Cut Temperature Sensor Folder.....	14
Figure 13: Short Cut Thermocoupler Wiring .....	15
Figure 14: Short Cut Wiring Diagram.....	15
Figure 15: Short Cut Outputs Tab .....	16
Figure 16: Short Cut Output Table Definition.....	17
Figure 17: Short Cut Compile Confirmation .....	17
Figure 18: PC200W Connect Button.....	18
Figure 19: PC200W Monitor Data Tab .....	19
Figure 20: PC200W Monitor Data Tab .....	19
Figure 21: PC200W Collect Data Tab.....	20
Figure 22: PC200W View Data Utility .....	20
Figure 23: PC200W View Data Table.....	21
Figure 24: PC200W View Data Graph.....	21
Figure 25: Features of a Data Acquisition System .....	24
Figure 26: CR200(X) Wiring Panel.....	31
Figure 27: Voltage Measurement Accuracy (0° to 40° C).....	40
Figure 28: Voltage Excitation Bridge Circuit.....	41
Figure 29: Switch Closure Pulse Sensor.....	42
Figure 30: Pulse Input Types.....	44
Figure 31: Amplitude Reduction of Pulse-Count Waveform (before and after 1 ms time constant filter).....	44
Figure 32: Current Limiting Resistor in a Tipping Bucket Rain Gage Circuit.....	46
Figure 33: Control Port Current Sourcing .....	50
Figure 34: Relay Driver Circuit with Relay .....	51
Figure 35: Power Switching without Relay .....	51
Figure 36: Lightning Protection Scheme .....	57
Figure 37: DevConfig Utility .....	60
Figure 38: DevConfig OS Download Window .....	61
Figure 39: Dialog Box Confirming OS Download.....	61
Figure 40: DevConfig Settings Editor .....	62
Figure 41: Summary of CR200(X) Configuration.....	63
Figure 42: DevConfig Deployment Tab .....	64
Figure 43: DevConfig Logger Control Tab .....	65
Figure 44: Entering SDI-12 Transparent Mode.....	117
Figure 45: Input Sample Vectors.....	121
Figure 46: Mean Wind Vector.....	123
Figure 47: Standard Deviation of Direction .....	124
Figure 48: Data from TrigVar Program.....	126

Figure 49: PakBus Network Addressing .....	134
Figure 50: Flat Map.....	138
Figure 51: Tree Map.....	138
Figure 52: Enclosure .....	148
Figure 53: Accuracy, Precision, and Resolution .....	Ap. 13

## List of Tables

Table 1. CR200 series Dataloggers with Built-In Radio .....	2
Table 2. PC200W EZSetup Wizard Example Selections .....	12
Table 3. Internal Lithium Battery Specifications .....	34
Table 4. Current Sourcing Limits.....	38
Table 5. Formats for Entering Numbers in CRBASIC.....	71
Table 6. CRBASIC Program Structure .....	72
Table 7. Predefined Constants and Reserved Words.....	77
Table 8. TOA5 Environment Line .....	78
Table 9. Typical Data Table.....	79
Table 10. Rules for Names.....	85
Table 11. Logical Expression Examples .....	89
Table 12. Abbreviations of Names of Data Processes.....	90
Table 13. Derived Trigonometric Functions .....	101
Table 14. Standard SDI-12 Command & Response Set.....	113
Table 15. Example Power Usage Profile for a Network of SDI-12 Probes.....	119
Table 16. OutputOpt Options.....	120
Table 17. CR200(X) Telecommunications Options .....	131
Table 18. PakBus Link Performance Gage .....	137
Table 19. Modbus to Campbell Scientific Equivalents .....	139
Table 20. CRBASIC Ports, Flags, Variables and Modbus Registers .....	141
Table 21. LoggerNet Products that Include the LoggerNet Server .....	144
Table 22. LoggerNet Clients .....	145
Table 23. Internal Lithium Battery Specifications .....	149
Table 24. Program Download Errors .....	152
Table 25. Math Expressions and CRBASIC Results.....	154
Table 26. Status Table Fields and Descriptions .....	Ap. 16
Table 27. CR200(X) Settings.....	Ap. 18
Table 28. CR200(X) RS-232 Pin-Out .....	Ap. 21

## List of CRBasic Examples

CRBASIC EXAMPLE 1. Inserting Comments.....	69
CRBASIC EXAMPLE 2. Load binary information into a single variable.....	71
CRBASIC EXAMPLE 3. Proper Program Structure .....	73
CRBASIC EXAMPLE 4. Using a variable array in calculations.....	75
CRBASIC EXAMPLE 5. Flag Declaration and Use .....	76
CRBASIC EXAMPLE 6. Using the Const Declaration.....	76
CRBASIC EXAMPLE 7. Alias and Unit Declaration .....	77
CRBASIC EXAMPLE 8. Definition and Use of a Data Table .....	80
CRBASIC EXAMPLE 9. Use of the Disable Variable.....	82
CRBASIC EXAMPLE 10. Use of a Subroutine .....	83
CRBASIC EXAMPLE 11. BeginProg / Scan / NextScan / EndProg Syntax.....	84
CRBASIC EXAMPLE 12. Measurement Instruction Syntax .....	84
CRBASIC EXAMPLE 13. Use of Expressions in Parameters.....	86
CRBASIC EXAMPLE 14. Use of Arrays as Multipliers and Offsets .....	86
CRBASIC EXAMPLE 15. Use of Variable Arrays to Conserve Code Space.....	88
CRBASIC EXAMPLE 16. Example Wireless Sensor Program For CR200(X) .....	109

CRBASIC EXAMPLE 17. CRBASIC EXAMPLE. Radio Power Minimization  
Program Examples ..... 111

CRBASIC EXAMPLE 18. CRBASIC EXAMPLE. Two Rain Gages on a  
CR200(X)..... 112

CRBASIC EXAMPLE 19. Using TrigVar to Trigger Data Storage ..... 126

CRBASIC EXAMPLE 20. CRBASIC EXAMPLE. Programming for two data  
intervals in one data table..... 127

CRBASIC EXAMPLE 21. Using NAN in Expressions..... 153

# Section 1. Introduction

---

Whether in extreme cold in Antarctica, scorching heat in Death Valley, salt spray from the Pacific, micro-gravity in space, or the harsh environment of your office, Campbell Scientific dataloggers support research and operations all over the world. Our customers work a broad spectrum of applications, from those more complex than any of us imagined, to those simpler than any of us thought practical. The limits of the CR200(X) are defined by our customers. Our intent with the CR200(X) manual is to guide you to the tools you need to explore the limits of your application.

You can take advantage of the CR200(X)'s powerful analog and digital measurement features by spending a few minutes working through the [Quickstart Tutorial](#) (p. 3) and the [Overview](#) (p. 23). For more demanding applications, the remainder of the manual and other Campbell Scientific publications are available. If you are programming with CRBASIC, you will need the extensive Help available with the CRBASIC Editor software.

This manual is organized to take you progressively deeper into the complexity of CR200(X) functions. You may not find it necessary to progress beyond the [Quickstart Tutorial](#) (p. 3) or [Overview](#) (p. 23) sections. [Quickstart Tutorial](#) (p. 3) gives a cursory view of CR200(X) data acquisition and walks you through a first attempt at data acquisition. [Overview](#) (p. 23) reviews salient topics, which are covered in-depth in subsequent sections and appendices.

More in-depth study requires other Campbell Scientific publications, most of which are available on-line at [www.campbellsci.com](http://www.campbellsci.com). Generally, if a particular feature of the CR200(X) requires a peripheral hardware device, more information is available in the manual written for that device. Manuals for Campbell Scientific products are available at [www.campbellsci.com](http://www.campbellsci.com).

If you are unable to find the information you need, please contact us at 435-753-2342 and speak with an applications engineer. Or you can email us at [support@campbellsci.com](mailto:support@campbellsci.com).

## 1.1 CR200(X) series Datalogger Models

Models CR200X and CR200 Dataloggers do not have a built-in spread spectrum radio.

The CR206X, CR211X, and CR216X combine the CR200X datalogger with a spread spectrum radio for telemetering data. The different model numbers are for different spread spectrum frequency ranges:

Table 1. CR200 series Dataloggers with Built-In Radio		
Model	Frequency	Where Used
CR206X CR206 (retired) CR205 (retired)	910 to 918 MHz	U.S./Canada
CR211X CR211 (retired) CR210 (retired)	920 to 928 MHz	Australia/Israel
CR216X CR216 (retired) CR215 (retired)	2.450 to 2.482 GHz	Worldwide

---

**Caution** No product using the 24XStream radio, including CR216X, will be available for sale in Europe after 1/1/2015 due to changes in EU legislation. Consequently, purchase of the CR216X is not recommended for use in Europe in new networks that may require future expansion.

---

The CR295X and CR295 (retired) GOES Dataloggers include an additional 9-pin serial port that allow communications with a TX320, the retired TX312, or the retired SAT HDR GOES satellite transmitter. While the CR295 required a special operating system, the CR295X does not.

Note: Throughout this manual CR200(X) will be used to refer to all of the different models of datalogger in the CR200-series and CR200X-series. In the cases where information applies only to a specific model or series of datalogger, that will be clearly specified.

The CR200(X)-series dataloggers have the following enhanced features as compared to the CR200-series dataloggers:

1. 128 Public variables can be used (CR200-series had 48).
2. 8 Data Tables can be declared (CR200-series had 4).
3. Compiled CRBasic program can be two times larger than for CR200-series.
4. All CR200(X)-series CRBasic instructions are supported in a single operating system. See CRBasic and CRBasic Help for a list of available instructions. No new instructions have been added. If new instructions are added in the future, they will apply only to the CR200(X)-series dataloggers.



## Section 2. Quickstart Tutorial

---

Quickstart tutorial gives a cursory look at CR200(X) data acquisition.

### 2.1 Primer - CR200(X) Data Acquisition

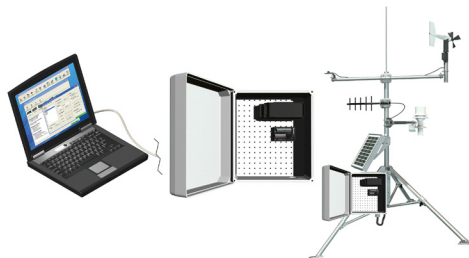
Data acquisition with the CR200(X) is the result of a step wise procedure involving the use of electronic sensor technology, the CR200(X), a telecommunications link, and PC datalogger support software.

#### 2.1.1 Components of a Data Acquisition System

A typical data acquisition system is conceptualized in [FIGURE. Data Acquisition System Components](#) (p. 3). A CR200(X) is only one part of a data acquisition system. To acquire good data, suitable sensors and a reliable data retrieval method are required. A failure in any part of the system can lead to "bad" data or no data.

##### 2.1.1.1 How Programmed Instructions Are Evaluated

The CR200(X) evaluates programmed instructions sequentially.



*Figure 1: Data Acquisition System Components*

##### 2.1.1.2 Sensors

Suitable sensors accurately and precisely transduce environmental change into measurable electrical properties by outputting a voltage, changing resistance, outputting pulses, or changing states.

---

**Read More! APPENDIX. Accuracy, Precision, and Resolution** (Appendix p. 13)

---

### 2.1.1.3 Datalogger

CR200(X)s can measure most sensors with an electrical response. CR200(X)s measure electrical signals and convert the measurement to engineering units, perform calculations and reduce data to statistical values. Every measurement does not need to be stored. The CR200(X) will store data in memory awaiting transfer to the PC via external storage devices or telecommunications.

### 2.1.1.4 Data Retrieval

The main objective of a data acquisition system is to provide data files on a PC.

Data are copied, not moved, from the CR200(X) to the PC. Multiple users may have access to the same CR200(X) without compromising data or coordinating data collection activities.

A RS-232 port is integrated with the CR200(X) wiring panel to facilitate data collection.

On-site serial communications are preferred if the datalogger is near the PC, and the PC can dedicate a serial (COM) port for the datalogger or use a USB-to-serial converter. On-site methods such as direct serial connection or infrared link are also used when the user visits a remote site with a laptop or PDA.

In contrast, telecommunications provide remote access and the ability to discover problems early with minimum data loss. Typically a base station radio that is compatible with the radio internal to the CR200(X) will be the preferred method of telecommunication. A variety of devices, such as telephone modems, satellite transceivers, and TCP/IP network modems may be interfaced with the base station for the most demanding applications.

## 2.1.2 CR200(X) Mounting

The CR200(X) module integrates electronics within a compact housing, making it economical, small, and very rugged.

## 2.1.3 Wiring Panel

As shown in [FIGURE. CR200\(X\) Wiring Panel](#) p. 5, the CR200(X) provides terminals for connecting sensors, power and communications devices. Internal surge protection is incorporated with the input channels.

The CR200(X) uses spring-loaded terminal blocks for connecting sensors and peripherals. This provides quick, vibration resistant connections. To attach wires, insert a small flat blade screwdriver into the slot and push back. Insert the wire and then bring the screwdriver forward.

---

Caution! Opening a terminal by prying the end may cause damage, particularly at low temperatures.

---

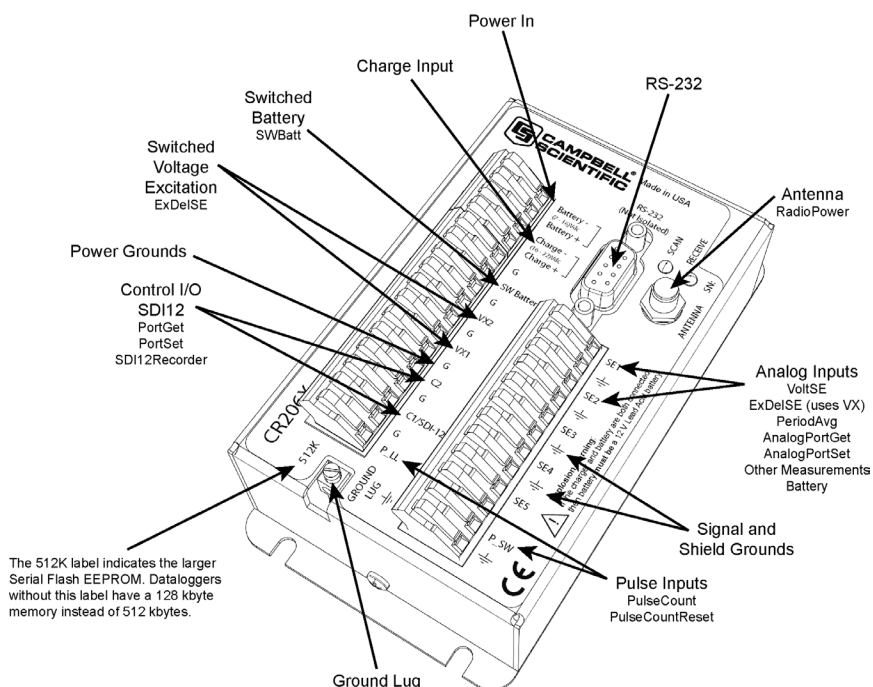


Figure 2: CR200(X) Wiring Panel

### 2.1.4 Battery Backup

A lithium battery backs up the CR200(X) clock, program, and memory if it loses power.

### 2.1.5 Power Supply

The CR200(X) is powered by a nominal 12 volt DC source. Acceptable power range is 7 to 16 VDC.

The CR200(X) does not have an internal power supply but does have connections for an external battery and a built-in charging regulator for charging a 12 V lead-acid battery from an external power source. Charging power can come from a 16-22 VDC input such as a solar panel.

### 2.1.6 Antenna

For CR200(X) models with a built-in radio, an FCC authorized antenna is a required component. An SMA male connector is provided on the CR200(X) wiring panel for antenna connection. Antennas are either 900 MHz or 2.4 GHz depending on the type of radio installed.

## 2.1.7 Analog Sensors

Analog sensors output continuous voltages that vary with the phenomena measured.

Analog sensors connect to analog terminals. Analog terminals are configured as single-ended, wherein sensor outputs are measured with respect to ground (*FIGURE. Analog Sensor Wired to Single-Ended Channel #1* (p. 6)). The CR200(X) cannot perform differential voltage measurements.

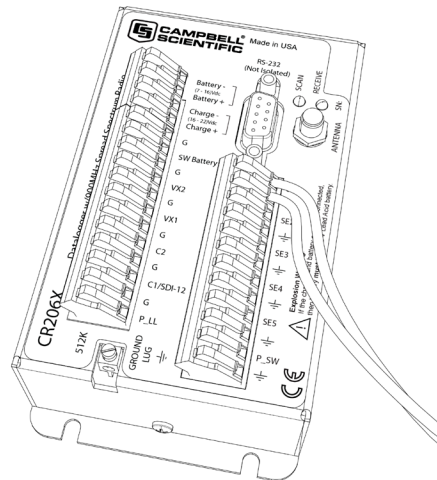


Figure 3: Analog Sensor Wired to Single-Ended Channel #1

## 2.1.8 Bridge Sensors

Bridge sensors change resistance with respect to environmental change. Resistance is determined by measuring the difference between the excitation voltage supplied to the bridge and the voltage detected by the CR200(X) returning from the bridge.

### 2.1.8.1 Voltage Excitation

The CR200(X) supplies a precise excitation voltage via excitation terminals. Return voltage is measured on single ended analog terminals. Because the CR200(X) cannot make the differential voltage readings used with full bridge and some half bridge circuits, only basic half bridge measurements can be made. A wiring example is illustrated in *FIGURE. Half Bridge Wiring Wind Vane Potentiometer* p. 7.

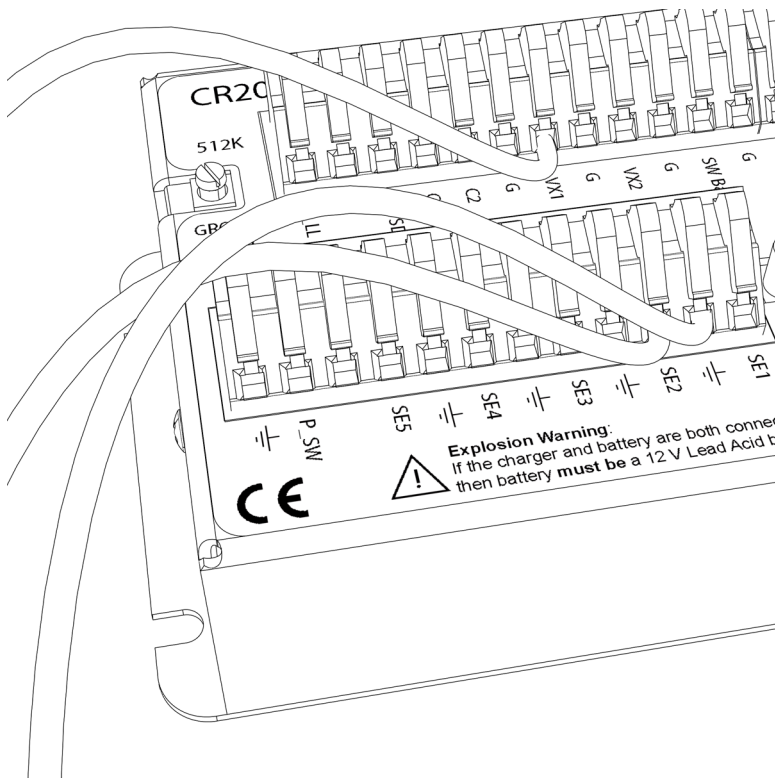


Figure 4: Half Bridge Wiring -- Wind Vane Potentiometer

### 2.1.9 Pulse Sensors

The CR200(X) can measure switch closures, low-level AC signals (waveform breaks zero volts), or voltage pulses. Compatible signal types are illustrated in [FIGURE. Pulse Input Types](#) (p. 7). A pulse input wiring example is shown in [FIGURE. Pulse Input Wiring -- Anemometer Switch](#) (p. 8).

---

**Note** Period averaging sensors are connected to analog channels.

---

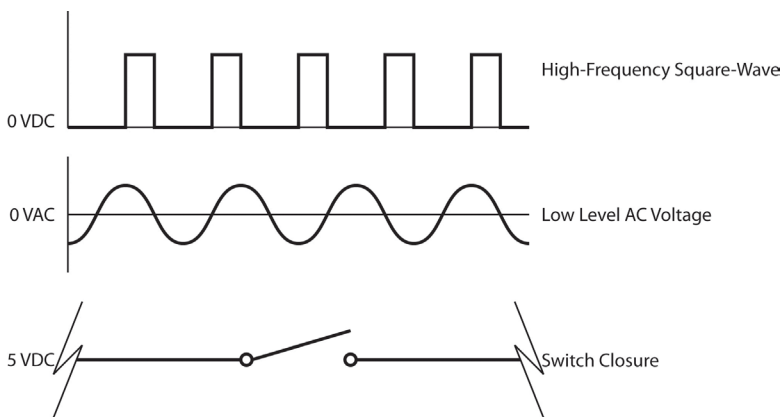


Figure 5: Pulse Input Types

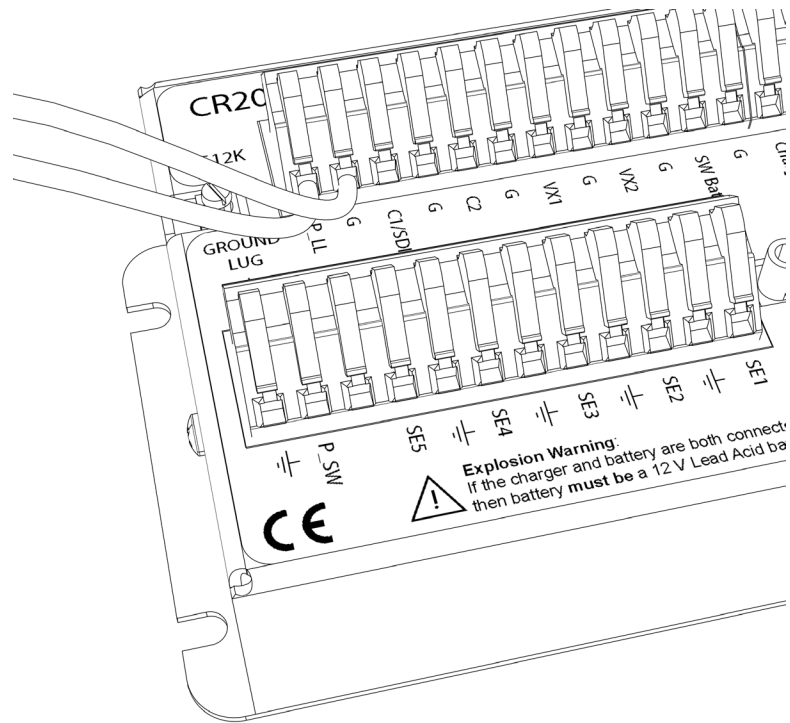


Figure 6: Pulse Input Wiring -- Anemometer Switch

### 2.1.10 Digital I/O Ports

The CR200(X) has 2 digital I/O ports selectable, under program control, as binary inputs or control outputs. These are multi-function ports including: device driven interrupts, switch closure pulse counting, high frequency pulse counting, and SDI-12 communications. [FIGURE. Control and Monitoring with Digital I/O](#) (p. 9), illustrates a simple application wherein a port is used to control a device while a second port monitors the state of the device.

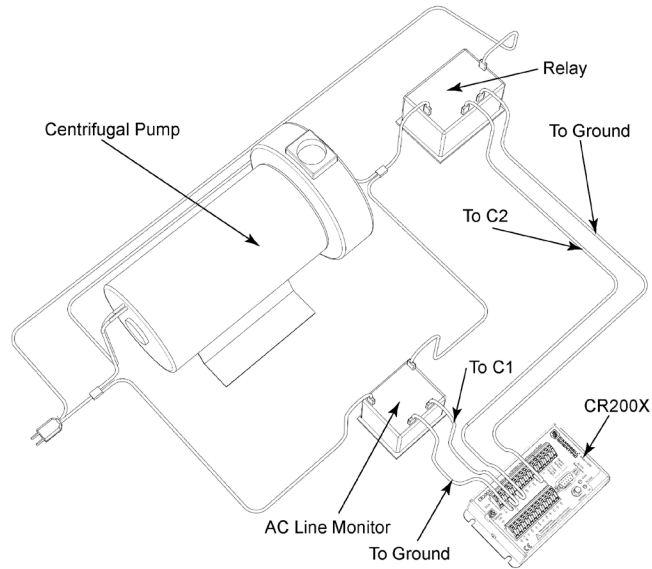


Figure 7: Control and Monitoring with Digital I/O

### 2.1.11 RS-232 Sensors

The CR200(X) has an RS-232 input as shown in [FIGURE. Location of RS-232 Port](#) p. 10. As indicated in [FIGURE. Use of RS-232 when Reading RS-232 Devices](#), p. 10 RS-232 sensors can be connected to the RS-232 port. The port can be set up with various baud rates, parity options, stop bit options, and so forth as defined in CRBASIC Help.

---

Note: For the CR200, SerialInput () is a special instruction, which is available only in the special S operating system.

---

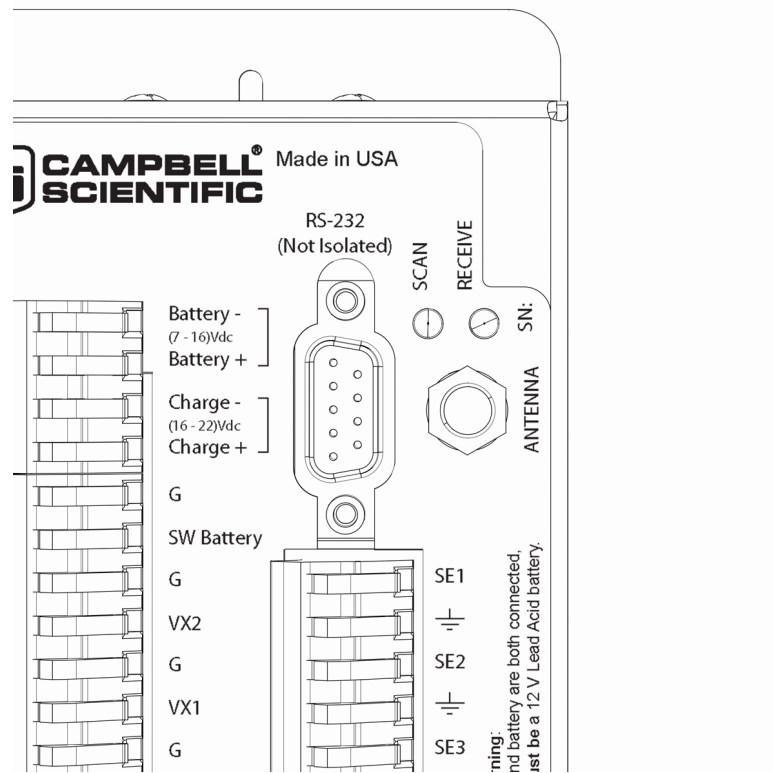


Figure 8: Location of RS-232 Port

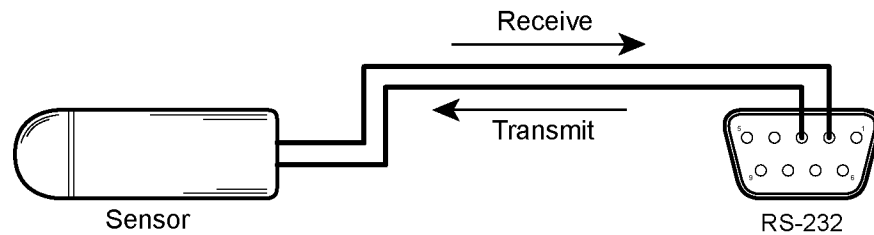


Figure 9: Use of RS-232 when Reading RS-232 Devices

## 2.2 Hands-on Exercise - Measuring Temperature

This tutorial is designed to illustrate the function of the CR200(X). During the exercise, the following items will be described.

- Attaching a temperature probe to analog differential terminals
- Creating a program for the CR200(X)
- Making a simple temperature measurement
- Sending data from the CR200(X) to a PC
- Viewing the data from the CR200(X)



## 2.2.1 Hardware Setup

With Reference to [FIGURE. Power and RS-232 Connections](#) (p. 11).

1. Connect external power (7 – 16VDC) to the CR200 by inserting the positive lead into the "Battery +".
2. Insert the negative lead into the "Battery-".
3. Connect the RS-232 cable (PN 10873, provided) between the RS-232 port on the CR200(X) and the RS-232 port on the PC. For computers that have only a USB port, a USB Serial Adaptor (PN 17394 or equivalent) is required.

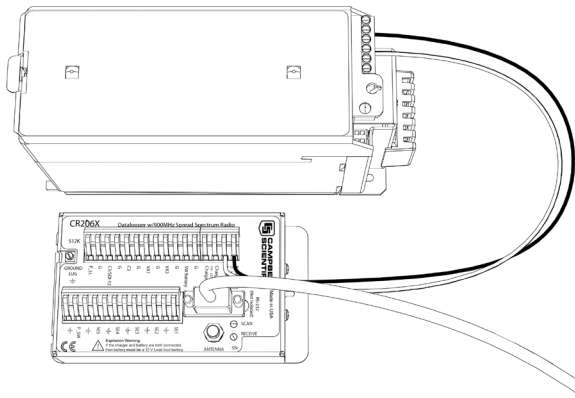


Figure 10: Power and RS-232 Connections

## 2.2.2 Configuration

For this exercise, factory default settings will work. To change the PakBus address or radio settings from their factory defaults, or if you are not sure what settings are currently stored on the CR200(X), use Device Configuration Utility or DevConfig software.

---

Read More! See [DevConfig](#) p. 59

---

## 2.2.3 PC200W Software Setup

1. Install the PC200W software onto a PC. Follow the on-screen prompts during the installation process for the Program Folder and Destination Location.
2. Open the PC200W software ([FIGURE. PC200W Main Window](#) (p. 13)). When the software is first run, the EZSetup Wizard will be run automatically in a new window. This will configure the software to communicate with the CR200(X). [TABLE. PC200W EZSetup Wizard](#)

*Example Selections* (p. 12) indicates what information needs to be entered on each screen. Click on Next at the bottom of the screen to advance to the next screen.

<b>Table 2. PC200W EZSetup Wizard Example Selections.</b>	
Start the wizard to follow table entries	
<b>Screen Name</b>	<b>Information Needed</b>
Introduction	Provides an introduction to the EZSetup Wizard along with instructions on how to navigate through the wizard.
Datalogger Type and Name	Select the CR200(X) from the scroll window.  Accept the default name of "CR200(X)."
COM Port Selection	Select the correct COM port for RS-232 connection. Typically, this will be COM1. Other COM numbers are possible, especially when using a USB to serial cable.  Leave the COM Port Communication Delay at "00 seconds."  Note: When using a USB to serial cable, the COM number may change if the cable is moved to a different USB port. This will prevent data transfer between the software and CR200(X). Should this occur, simply move the USB cable back to the original port. If this is not possible, it will be necessary to close the PC200W software and open it a second time to refresh the available COM ports. Click on "Edit Datalogger Setup" and change the COM port to the new port number.
Datalogger Settings	Used to configure how the CR200(X) communicates through the COM port.  For this tutorial, accept the default settings.
Communication Setup Summary	Provides a summary of the settings made in previous screens.
Communications Test	A communications test between the CR200(X) and PC can be performed in this screen.  For this tutorial, the test is not required. Press Finish to exit the Wizard.

After exiting the wizard, the main PC200W window becomes visible. The window has several tabs available. By Default, the Clock/Program tab is visible. This tab displays information on the currently selected datalogger along with clock and program functions. The Monitor Data or Collect Data tabs may be selected at any time.

A number of icons are available across the top of the window. These access additional functions available to the user.

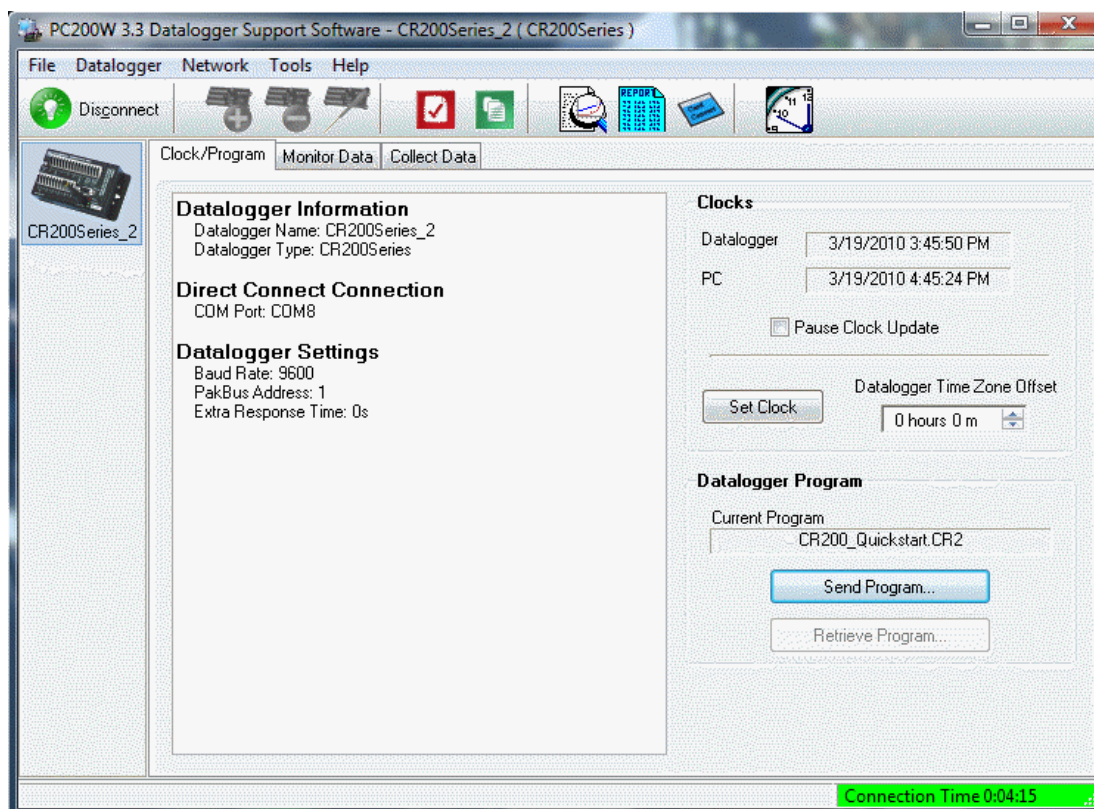


Figure 11: PC200W Main Window

### 2.2.3.1 Programming With Short Cut

#### 2.2.3.1.1 Short Cut Programming Objectives

This portion of the tutorial will use Short Cut to create a program that measures air temperature ( $^{\circ}\text{C}$ ) with a 109 Temperature Probe, and rainfall (mm) with a TE525WS rain gage. The CR200(X) will take samples every ten seconds and store averages of these values at one minute intervals.

Even if the 109 Temperature Probe and TE525WS Rain Gage sensors are not available, the programming example can still be followed. Without a 109 probe connected the measurement result will be NAN; without a TE525WS connected the measurement result will be 0. A rain gage can be simulated by straightening a segment of each of two paper clips and inserting the straightened segment of one paper clip into P\_SW and the adjacent ground channel. To simulate a rain gage tip, squeeze the paper clips together until they touch and then allow them to spring apart.

#### 2.2.3.1.2 Procedure (Short Cut Steps 1–6)

1. Click on the Short Cut icon in the upper-right corner of the PC200W window. The icon resembles a clock face.

2. A new window will appear showing the option to create a new program or open an existing program. Select New Program.
3. A drop-down list will appear showing different dataloggers. Select the CR200(X) and click OK.
4. The program will now ask for the scan interval. Set the interval to 10 seconds and click OK.
5. A second prompt will ask for a choice of "Sensor Support." Select "Campbell Scientific, Inc."
6. Under Available Sensors, expand the "Sensors" folder by clicking on the "+" symbol. This shows several sub-folders. Expand the "Temperature" folder to view the available sensors.

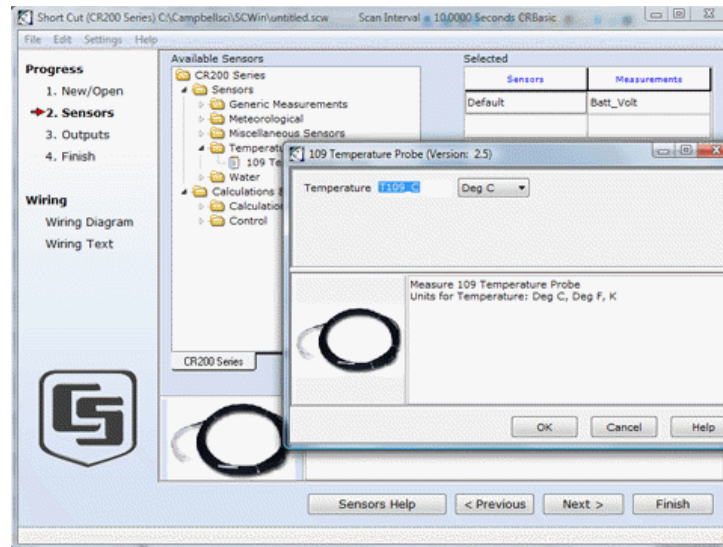


Figure 12: Short Cut Temperature Sensor Folder

### 2.2.3.1.3 Procedure (Short Cut Steps 7–9)

1. Double-click the 109 Temperature Probe sensor to add it to the Selected category. Alternatively, highlight the Wiring Panel Temperature sensor by clicking on it once, and then click on the arrow between Available Sensors and Selected to add it to the Selected sensors.
2. Click **OK** on the next screen to accept T109\_C for the measurement label, the DegC for the units.
3. Double click on the **Meteorological** application group. Double click on Precipitation, and double click on the **TE525 / TE525WS** sensor to add it to the selected sensors table. Click OK to accept Rain\_mm for the measurement label, and mm for the units.

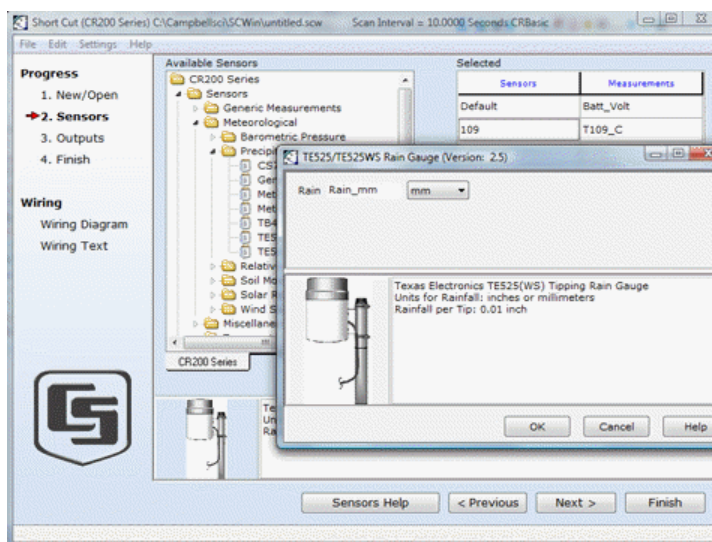


Figure 13: Short Cut Thermocouple Wiring

#### 2.2.3.1.4 Procedure (Short Cut Step 10)

1. Click on the Wiring Diagram link to view the sensor wiring diagram. Attach the 109 Temperature Probe and TE525 Rain Gauge to the CR200(X) as shown in the diagram. Click on Outputs to advance to the next step.

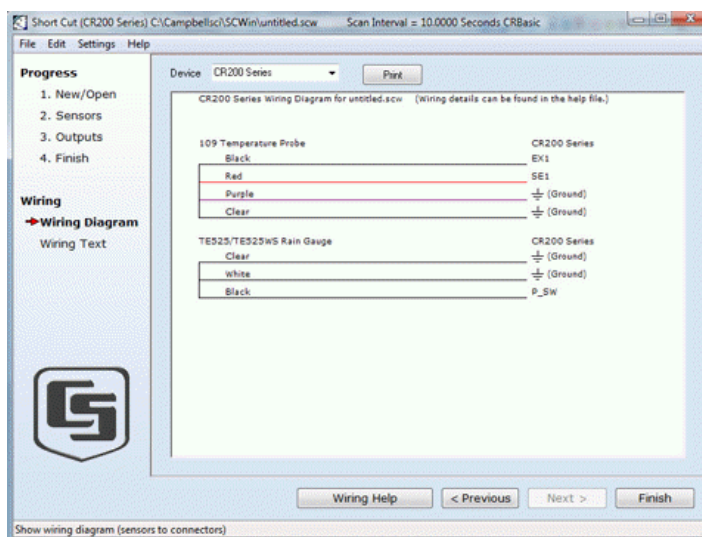


Figure 14: Short Cut Wiring Diagram

#### 2.2.3.1.5 Procedure (Short Cut Step 11)

1. The Outputs window displays a list of selected sensors on the left, and data storage Tables on the right.

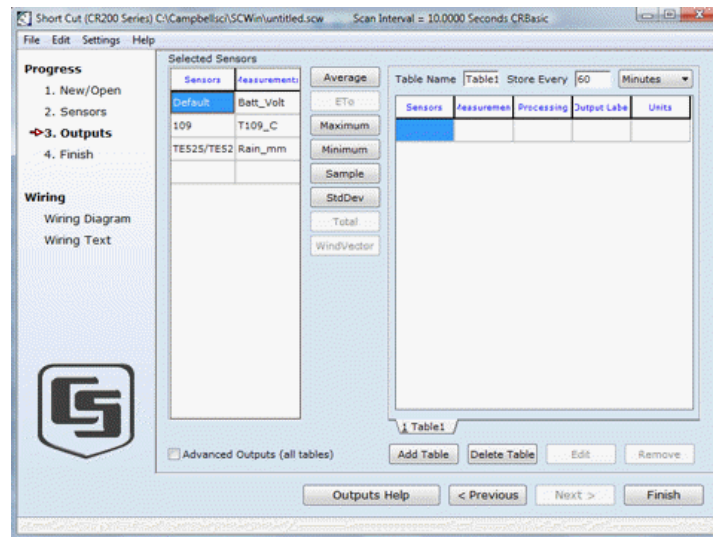


Figure 15: Short Cut Outputs Tab

#### 2.2.3.1.6 Procedure (Short Cut Steps 12 –18)

1. By default, there are two Tables initially available. Both Tables have a Store Every field along with a drop-down box to select the time units. These are used to set the time interval when data is stored.
2. Only one Table is needed for this tutorial, so Table 2 can be removed. Select Table 2 by clicking on its tab, and then click on Delete Table.
3. Change the Table Name to OneMin, and then change the interval to 1 minute (Store Every 1 Minutes).
4. Adding a measurement to the table is done by selecting the measurement under Selected Sensors, and then clicking on one of the processing buttons in the center of the window.
5. Click the Default sensor (battery voltage) and click the Minimum button. Click the 109 temperature sensor and click the Average button. Click the TE525 rain gauge sensor and click the Total button.
6. Click the Default sensor (battery voltage) and double click the Minimum button. Click the 109 temperature sensor and double click the Average button. Click the TE525 rain gauge sensor and double click the Total button.
7. Click the Default sensor (battery voltage) and double click the Minimum button. Do not store the time of minimum. Click the 109 temperature sensor and double click the Average button. Click the TE525 rain gauge sensor and double click the Total button..



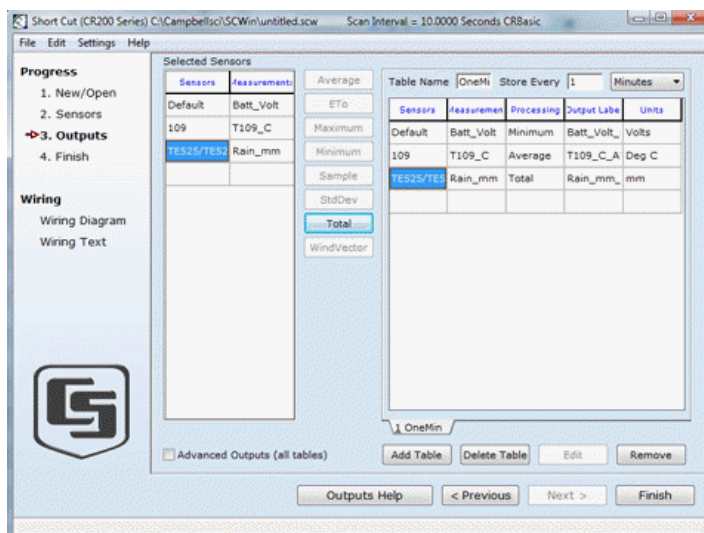


Figure 16: Short Cut Output Table Definition

### 2.2.3.1.7 Procedure (Short Cut Step 19)

1. Click on Finish to compile the program. Give the program the name "QuickStart." A prompt will ask if you want to send the program to the datalogger. For this exercise choose No. A summary screen will appear showing the compiler results. Any errors during compiling will also be displayed.

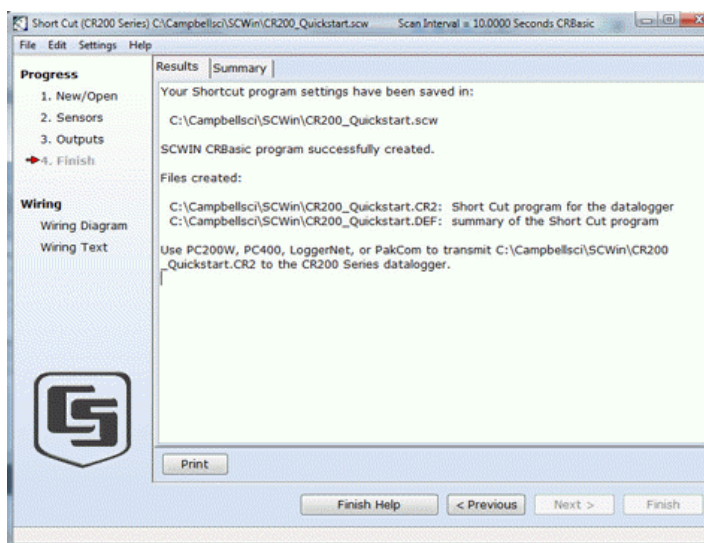


Figure 17: Short Cut Compile Confirmation

### 2.2.3.1.8 Procedure (Short Cut Step 20)

1. Close this window by clicking on the "X" in the upper right corner.

### 2.2.3.2 Programming the CR200(X) and Collecting Data

#### 2.2.3.2.1 Procedure (PC200W Step 1)

1. From the PC200W Clock/Program tab, click on the *Connect* button to establish communications with the CR200(X). When communications have been established, the text on the button will change to Disconnect.

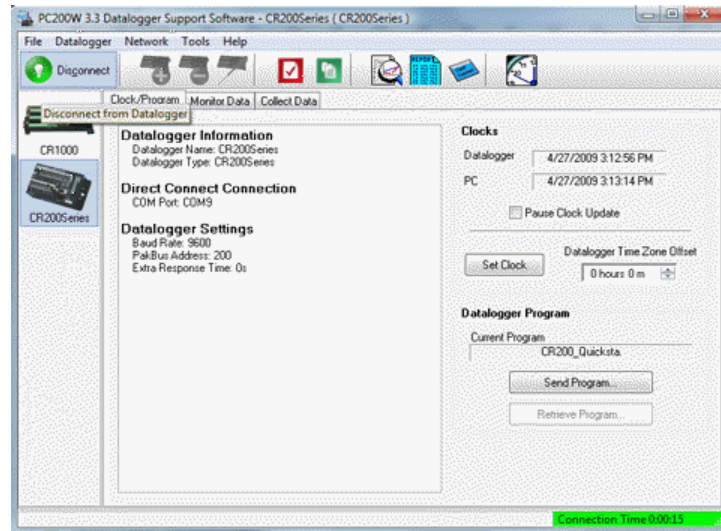


Figure 18: PC200W Connect Button

#### 2.2.3.2.2 Procedure (PC200W Steps 2–4)

1. Click the Set Clock button to synchronize the datalogger's clock with the computer's clock.
2. Click on the Send Program button. A window will appear warning that data on the datalogger will be erased. Answer "yes" to the prompt. Another window will open. Browse to the C:\CampbellSci\SCWin folder, select the **QuickStart.CR2** file, and then click the Open button. A status bar will appear while the program is sent to the CR200(X) followed by a confirmation that the transfer was successful. Click OK to close this window.
3. After sending a program to the CR200(X), a good practice is to monitor the measurements to ensure they are reasonable. Select the Monitor Data tab. The window now displays data found in the Public Table coming from the CR200(X). To view the OneMin table, select an empty cell in the display area, and then click on the *Add* button.



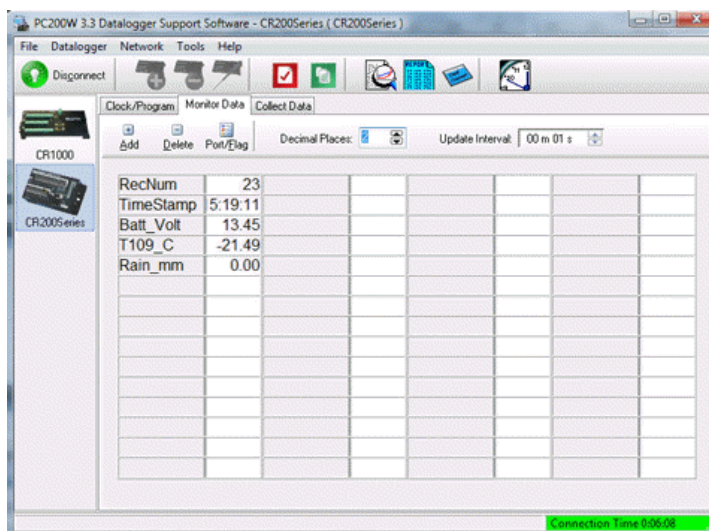


Figure 19: PC200W Monitor Data Tab

#### 2.2.3.2.3 Procedure (PC200W Step 5)

1. In the Add Selection window, click on the OneMin table, and then click Paste. The OneMin table is now displayed in the main display.

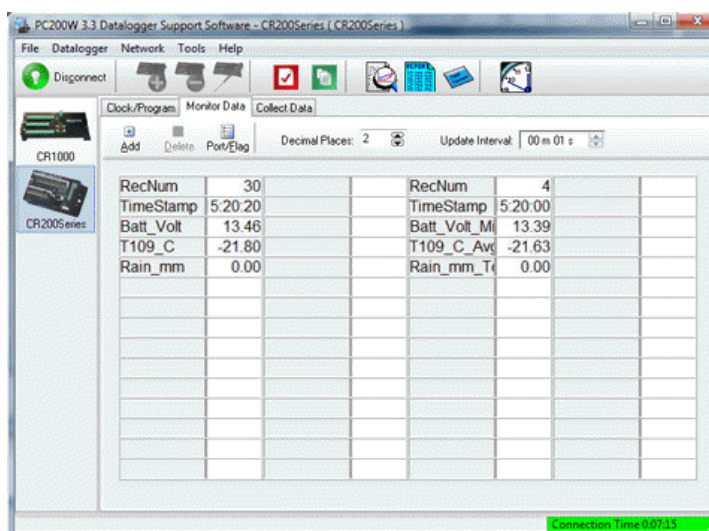


Figure 20: PC200W Monitor Data Tab

#### 2.2.3.2.4 Procedure (PC200W Step 6)

1. Click on the Collect Data tab. From this window, data is chosen to be collected as well as the location where the collected data will be stored.

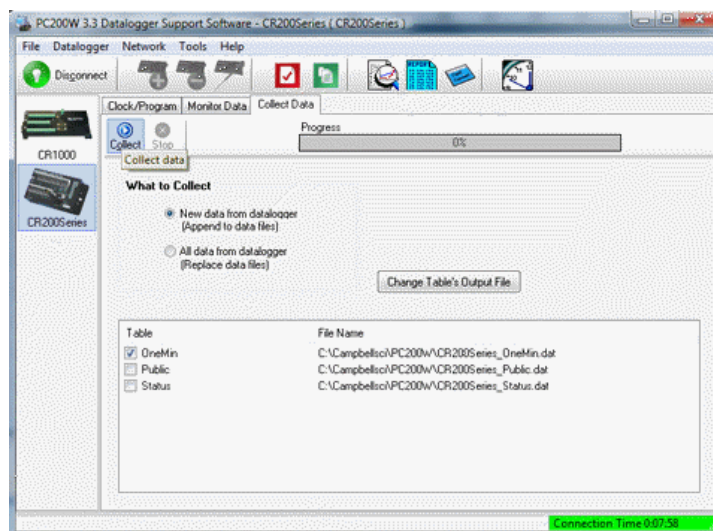


Figure 21: PC200W Collect Data Tab

#### 2.2.3.2.5 Procedure (PC200W Steps 7–9)

1. Click the OneMin box so a check mark appears in the box. Under the "What to Collect" heading, select "New data from datalogger." This selects which data will be collected.
2. Click on the Collect button. A progress bar will appear as the data is collected, followed by the message, "Collection Complete." Click **OK** to continue.
3. To view the data, click on the View icon at the top of the window. This opens a new window.

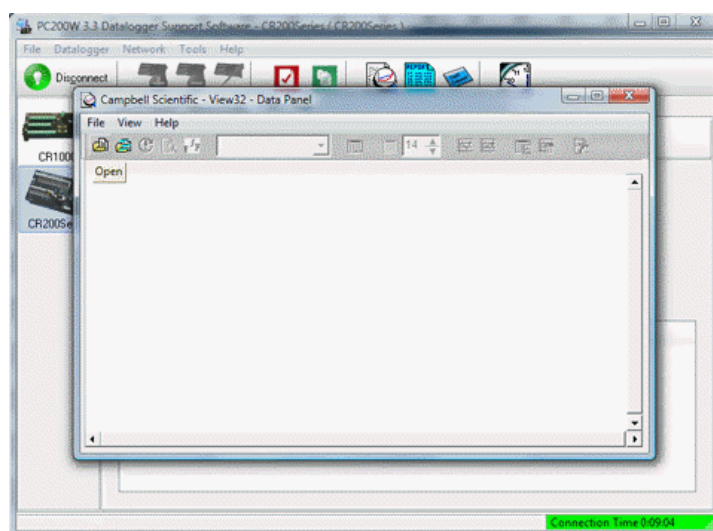
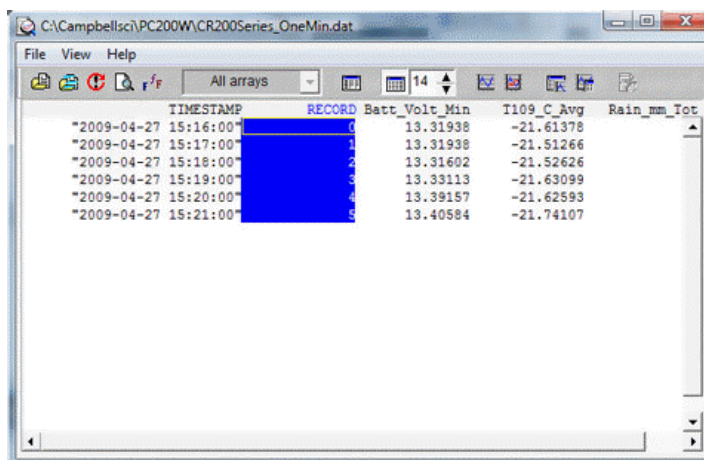


Figure 22: PC200W View Data Utility

**2.2.3.2.6 Procedure (PC200W Step 10)**

1. Click on the Open File icon to open a file for viewing. Select the "CR200Series\_OneMin.dat" file and click on Open. The collected data is now shown.



TIMESTAMP	RECORD	Batt_Volt_Min	T109_C_Avg	Rain_mm_Tot
"2009-04-27 15:16:00"	0	13.31938	-21.61378	
"2009-04-27 15:17:00"	1	13.31938	-21.51266	
"2009-04-27 15:18:00"	2	13.31602	-21.52626	
"2009-04-27 15:19:00"	3	13.33113	-21.63099	
"2009-04-27 15:20:00"	4	13.39157	-21.62593	
"2009-04-27 15:21:00"	5	13.40584	-21.74107	

Figure 23: PC200W View Data Table

**2.2.3.2.7 Procedure (PC200W Step 11)**

1. Select any data column by clicking on it. To display the data in graphical form, click on one of the Show Graph buttons. A graph with one Y-axis or two Y-axes will be generated.

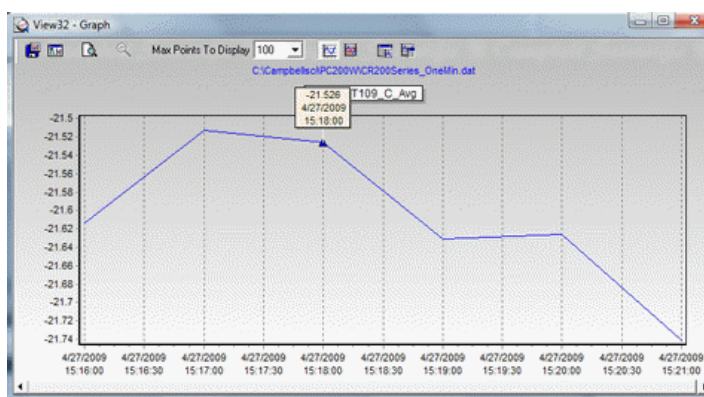


Figure 24: PC200W View Data Graph

**2.2.3.2.8 Procedure (PC200W Step 12)**

1. Close the Graph and View windows, and then close the PC200W program.



## Section 3. Overview

---

### 3.1 CR200(X) Overview

The CR200(X) Datalogger is a precision instrument designed for low-power measurement applications. CPU, analog inputs, digital outputs, and memory are controlled by the operating system in conjunction with the user program. The user program is written in CRBASIC, a programming language that includes data processing and analysis routines and a standard BASIC instruction set. Campbell Scientific's datalogger support software facilitates program generation, editing, data retrieval, and real-time data monitoring (see [Support Software](#) (p. 143)).

*FIGURE. Features of a Data Acquisition System* (p. 24) illustrates a common CR200(X)-based data acquisition system.

The CR200(X) is a multimeter with memory and timekeeping. It is one part of a data acquisition system. To acquire quality data, suitable sensors and reliable telecommunications devices are also required.

Sensors transduce phenomena into measurable electrical forms, outputting voltage, current, resistance, pulses, or state changes. The CR200(X), sometimes with the assistance of various peripheral devices, can measure most electronic sensors. Contact a Campbell Scientific applications engineer if you have questions about a sensor's compatibility with a CR200(X) datalogger.

The CR200(X) measures analog voltage and pulse signals, representing the magnitudes numerically. Numeric values are scaled to the unit of measure such as millivolts and pulses, or in user specified engineering units such as wind direction and wind speed. Measurements can be processed through calculations or statistical operations and stored in memory awaiting transfer to a PC via external storage or telecommunications.

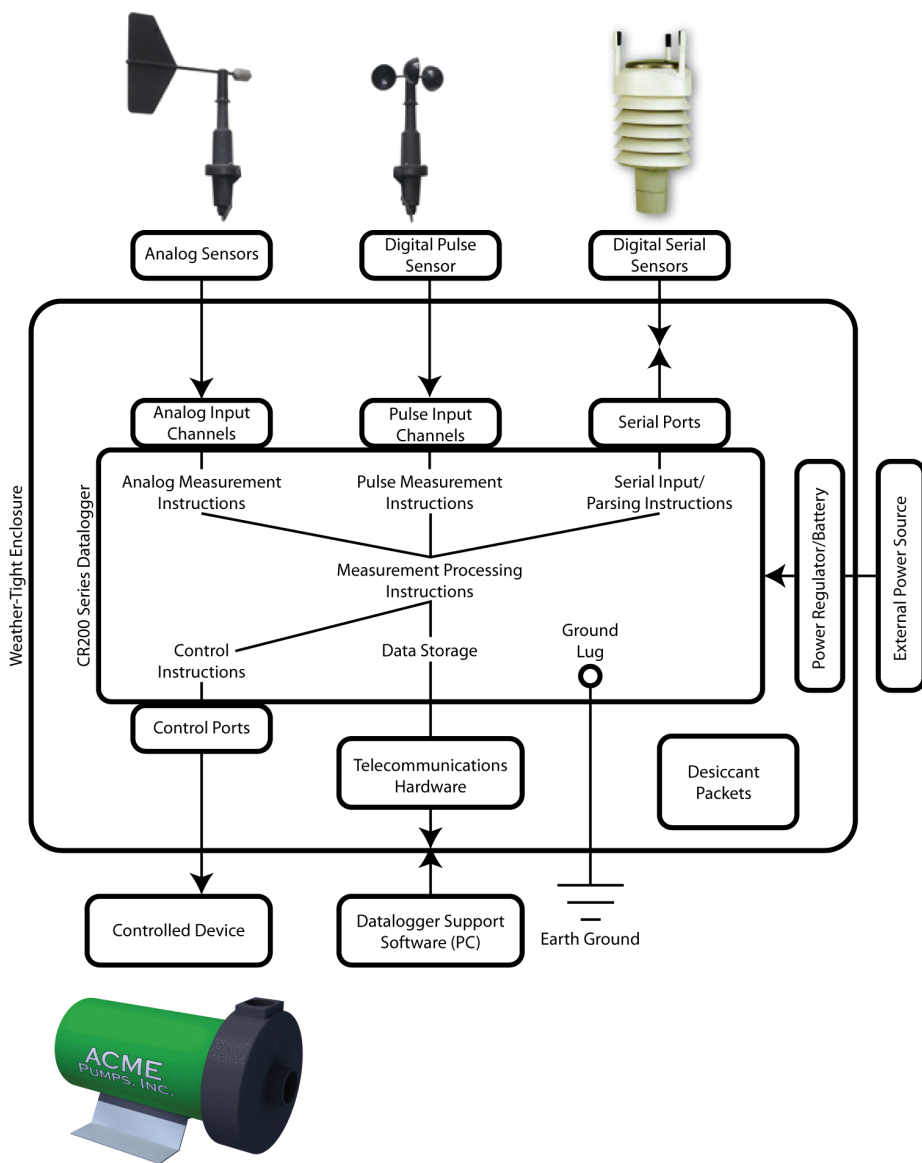


Figure 25: Features of a Data Acquisition System

### 3.1.1 Programmed Instructions Are Evaluated Sequentially

The CR200(X) evaluates programmed instructions sequentially.

### 3.1.2 Sensor Support

---

Read More! See *Sensor Support* (p. 37)

---

The following sensor types are supported by the CR200(X) datalogger.

- Analog voltage
- Analog current (with a shunt resistor)
- Resistive bridges
- Pulse output
- Period output
- Frequency output
- SDI-12 sensors

A library of sensor manuals and application notes are available at [www.campbellsci.com](http://www.campbellsci.com) to assist in measuring many sensor types. Consult with a Campbell Scientific applications engineer for assistance in measuring unfamiliar sensors.

### 3.1.3 Input / Output Interface: The Wiring Panel

The wiring panel of the CR200(X) is the interface to all CR200(X) functions. Most CR200(X) functions are best introduced by reviewing features of the CR200(X) wiring panel. *FIGURE. CR200(X) Wiring Panel* (p. 5) illustrates the wiring panel and some CR200(X) functions accessed through it.

---

**Read More!** Read *Measurement and Control Peripherals* (p. 49) for more information.

---

#### 3.1.3.1 Measurement Inputs

Measurements require a physical connection with a sensor at an input channel and CRBASIC programming to instruct the CR200(X) how to make, process, and store the measurement. The CR200(X) wiring panel has the following input channels:

**Analog Voltage:** 5 channels (SE 1–5).

- Input voltage range: 0 mV to 2500 mV.
- Measurement resolution: 0.6 mV

**Period Average:** 4 channels (SE 1–4)

- Maximum input voltage: 4000 mV.
- Maximum frequency: 150 kHz
- Voltage threshold: counts cycles on transition from < 900 mV to > 2100 mV.

---

**Note** Both pulse count and period average measurements are used to measure frequency output sensors. Yet pulse count and period average measurement methods are different. Pulse count measurements use dedicated hardware -- pulse count accumulators, which are always monitoring the input signal, even when the CR200(X) is between program scans. In contrast, period average measurement instructions only monitor the input signal during a program scan. Consequently, pulse count scans can usually be much less frequent than period average scans. Pulse counters may be more susceptible to low frequency noise because they are always "listening", whereas period averaging may filter the noise by reason of being "asleep" most of the time. Pulse count measurements are not appropriate for sensors that are powered off between scans, whereas period average measurements work well since they can be placed in the scan to execute only when the sensor is powered and transmitting a correct signal.

Period average measurements utilize a high-frequency digital clock to measure time differences between signal transitions, whereas pulse count measurements simply accumulate the number of counts. As a result, period average measurements offer much better frequency resolution per measurement interval, as compared to pulse count measurements. The frequency resolution of pulse count measurements can be improved by extending the measurement interval by increasing the scan interval and by averaging.

---

**Pulse:** 2 channels (P\_SW, P\_LL) configurable for counts or frequency of the following signal types:

- High level square waves (P\_SW)
- Switch closures (P\_SW)
- Low-level A/C sine waves (P\_LL)

**Digital I/O:** 2 channels (C1 - C2) configurable for SDI-12 communication, state, frequency, and pulses. 5 analog channels (SE1–SE5) configurable for input/output.

- C1 - C2: state, frequency, pulse.
- C1: SDI-12 input/output, state, frequency, pulse.
- SE 1–5: input/output, state.



### 3.1.3.2 Voltage Outputs

---

**Read More!** See [Control Output](#) (p. 49).

---

- **Switched Analog Output (Excitation):** two channels (VX1/VX2) for precise voltage excitation ranging from +2500 mV or +5000 mV. These channels are regularly used with resistive bridge measurements. Each channel will source up to 25 mA.
- **Digital I/O:** 2 channels (C1 - C2) configurable for on / off and pulse output duration.
- **Switched 12 Volts (SW Battery):** (switch on / off) primary voltage under program control for control of external devices requiring 12V DC, such as humidity sensors. SW Battery can source up to 900 mA. See [TABLE. Current Sourcing Limits](#) p. 38.
- **CR200(X) used as a PLC (programmable logic controller):** Utilizing peripheral relays and analog output devices, the CR200(X) can manage binary and variable control devices through the following output channels:
  - **Digital I/O:** 2 channels (C1/C2) configurable for pulse output duration.
  - **Switched 12 Volts (SW Battery):** (switches on/off) primary voltage under program control for control of external devices requiring 12V DC, such as humidity sensors. SW Battery can source up to 900 mA. See [TABLE. Current Sourcing Limits](#) p. 38.

### 3.1.3.3 Grounding Terminals

---

**Read More!** See [Grounding](#) (p. 55).

---

Proper grounding will lend stability and protection to a data acquisition system. It is the easiest and least expensive insurance against data loss-and the most neglected. The following terminals are provided for connection of sensor and datalogger grounding:

- **Signal Grounds:** 6 ground terminals (□) used as reference for single-ended analog inputs, pulse inputs, excitation returns, and as a ground for sensor shield wires. Signal returns for pulse inputs should use □ terminals located next to pulse inputs.
- **Power Grounds:** 6 terminals (G) used as returns for 5V, SW Battery, 12V, and C1-C2 outputs. Use of G grounds for these outputs minimizes potentially large current flow through the analog voltage measurement section of the wiring panel, which can cause single-ended voltage measurement errors.
- **Ground Lug:** 1 terminal (□), the large ground lug is used to connect a heavy gage wire to earth ground. A good earth connection is necessary to secure the ground potential of the datalogger and shunt transients away from electronics. Minimum 14 AWG wire is recommended.

### 3.1.3.4 Power Terminals

---

**Read More!** See [CR200\(X\) Power Supply](#) (p. 53).

---

#### Power In

- **Power Supply:** External battery is connected to Battery+ and Battery- terminals.
- Input voltages in excess of 18 V may damage the CR200 and/or power supply. Power to charge a 12 V lead-acid battery (16-22 VDC) must be connected to the Charge+ and Charge- terminals NOT to Battery+ and Battery-.

#### Power Out

- **Peripheral 12 V Power Source:** 1 terminal (SWBattery) and the associated ground (G) supply power to sensors and peripheral devices requiring nominal 12 VDC. Sensors and peripheral devices may also share a connection to the Battery+ and Battery- terminals which is the same as connecting them directly to the battery. This supply may drop as low as 7 VDC before datalogger operation stops. Precautions should be taken to minimize the occurrence of invalid data from underpowered sensors.

### 3.1.3.5 Communications Ports

---

**Read More!** See [Telecommunications and Data Retrieval](#) (p. 131) and [PakBus Overview](#) (p. 133).

---

The CR200(X) is equipped with an RS-232 communications port. This port allows the CR200(X) to communicate with other computing devices, such as a PC, or with other Campbell Scientific dataloggers.

---

**Note** RS-232 communications normally operate well up to a transmission cable capacitance of 2500 picofarads, or approximately 50 feet of commonly available serial cable.

---

- **9-pin RS-232:** Non-isolated port for communicating with a PC through the supplied serial cable, serial sensors, or through 3rd party serial telecommunications devices. Acts as a DTE device with a null-modem cable.

---

**Read More!** See [APPENDIX. Serial Port Pin Outs](#) (Appendix p. 21).

---

Some versions of the CR200(X) include an optional spread spectrum radio for wireless communication with other devices equipped with a spread spectrum radio.

### 3.1.4 Power Requirements

---

**Read More!** See [CR200\(X\) Power Supply](#) (p. 53).

---

The CR200(X) operates from a DC power supply with voltage ranging from 7 to 16 V, and is internally protected against accidental polarity reversal. The CR200(X) has modest input power requirements, typically an average current drain of less than 3 mA. Models with built-in radios may have a higher average current drain, depending on the radio's power mode and amount of time the radio is in use. Be sure to include the current usage of the radio and any powered sensors when calculating power supply requirements.

In low power applications, the CR200(X) can operate for several months on non-rechargeable batteries. Power systems for longer-term remote applications typically consist of a charging source, a charge controller, and a rechargeable battery. When AC line power is available, an AC/DC wall adapter and a rechargeable battery can be used to construct a UPS (uninterruptible power supply) in conjunction with the CR200(X)'s built-in voltage regulator/charge controller. Contact a Campbell Scientific applications engineer for assistance in acquiring the items necessary to construct a UPS.

Applications requiring higher current requirements, such as satellite or cellular phone communications, should be evaluated by means of a power budget with a knowledge of the factors required by a robust power system. Contact a Campbell Scientific applications engineer if assistance is required in evaluating power supply requirements.

Common power devices are:

- Batteries
  - Alkaline D-cell - 1.5 V/cell
  - Rechargeable Lead-Acid battery
- Charge Sources
  - Solar Panels
  - Wind Generators
  - AC/DC wall adapters

### 3.1.5 Programming: Firmware and User Programs

The CR200(X) is a programmable instrument, adaptable to demanding measurement and telecommunications requirements.

### 3.1.5.1 Firmware: OS and Settings

---

**Read More!** See [CR200\(X\) Configuration](#) (p. 59).

---

Firmware consists of the operating system (OS) and durable configuration settings. OS and settings remain intact when power is cycled.

---

**Note** The CR200(X) is shipped factory ready with all settings and firmware necessary to communicate with a PC via RS-232 and to accept and execute user application programs. OS upgrades are occasionally made available at [www.campbellsci.com](http://www.campbellsci.com).

---

For more complex applications, some settings may need adjustment. Adjustments are accomplished with DevConfig Software ([DevConfig](#) (p. 59)) or through datalogger support software (see [Support Software](#) (p. 143)).

OS files are sent to the CR200(X) with DevConfig or through the program Send button in datalogger support software. When the OS is sent via DevConfig, most settings are cleared, whereas, when sent via datalogger support software, most settings are retained.

### 3.1.5.2 User Programming

---

**Read More!** See [Programming](#) (p. 69) and [CRBASIC Programming Instructions](#) (p. 93) and CRBASIC help for more programming assistance.

---

A CRBASIC program directs the CR200(X) how and when sensors are to be measured, calculations made, and data stored. A program is created on a PC and sent to the CR200(X). Two Campbell Scientific software applications, Short Cut and CRBASIC Editor, create CR200(X) programs.

- Short Cut creates a datalogger program and wiring diagram in four easy steps. It supports most sensors sold by Campbell Scientific and is recommended for creating simple programs to measure sensors and store data.
- Programs generated by Short Cut are easily imported into CRBASIC Editor for additional editing. For complex applications, experienced programmers often create essential measurement and data storage code with Short Cut, then edit the code with CRBASIC Editor. Note that once a Short Cut generated program has been edited with CRBASIC Editor, it can no longer be modified with Short Cut.

## 3.1.6 Memory and Data Storage

---

**Read More!** See [Memory and Data Storage](#) (p. 129).

---

The CR200(X) has 2K SRAM for communication buffers, calculations, and variables, and 60K Flash EEPROM for the operating system and user program. The CR200 originally had a 128K Serial Flash EEPROM for data storage. Campbell Scientific has increased the data storage memory from 128 kbytes to

512 kbytes in newer CR200s and in all CR200(X)s. CR200s with the increased memory have "512K" on their label.

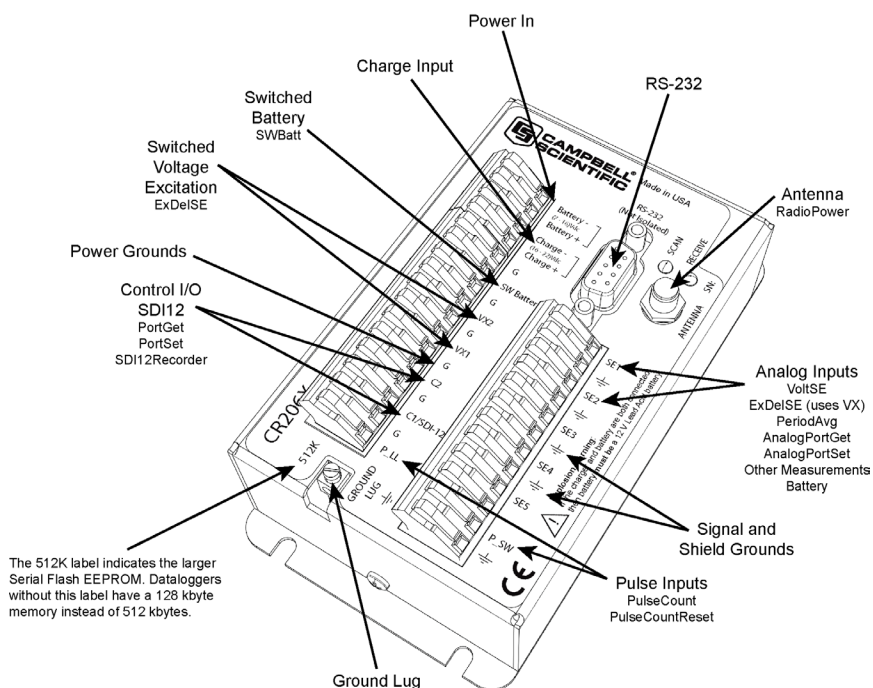


Figure 26: CR200(X) Wiring Panel

### 3.1.7 Communications Overview

---

**Read More!** See [Telecommunications and Data Retrieval](#) (p. 131).

---

The CR200(X) communicates with external devices to receive programs, send data, or act in concert with a network. The primary communication protocol is PakBus. Modbus is also supported.

#### 3.1.7.1 PakBus

---

**Read More!** See [PakBus Overview](#) (p. 133).

---

The CR200(X) communicates with Campbell Scientific support software, telecommunication peripherals, and other dataloggers via PakBus, a proprietary network communications protocol. PakBus is a protocol similar in concept to IP (Internet protocol). By using signed data packets, PakBus increases the number of communications and networking options available to the CR200(X). Communication can occur via RS-232 or RF.

Advantages of PakBus:

- Simultaneous communication between the CR200(X) and other devices.
- Peer-to-peer communication-no PC required.
- Other PakBus dataloggers can be used as "sensors" to consolidate all data into one CR200(X).
- Routing - the CR200(X) cannot act as a router, passing on messages intended for another logger, but other dataloggers can. PakBus supports automatic route detection and selection.
- Datalogger to datalogger communications-special CRBASIC instructions simplify transferring data between dataloggers for distributed decision making or control.
- In a PakBus network, each datalogger is set to a unique address before being installed in the network. Default PakBus address is 1. To communicate with the CR200(X), the datalogger support software (see [Telecommunications and Data Retrieval](#) (p. 131)) must know the CR200(X)'s PakBus address. The PakBus address is changed using the DevConfig software, CR200(X) status table, or PakBus Graph software.

### 3.1.7.2 Modbus

---

**Read More!** See [Modbus](#) (p. 139).

---

The CR200(X) supports Modbus Master and Modbus Slave communication for inclusion in Modbus SCADA networks.

## 3.1.8 Maintenance Overview

---

**Read More!** See Care and Maintenance.

---

With reasonable care, the CR200(X) should give many years of reliable service.

### 3.1.8.1 Protection from Water

The CR200(X) and most of its peripherals must be protected from moisture. Moisture in the electronics will seriously damage, and probably render un-repairable, the CR200(X). Water can come from flooding or sprinkler irrigation, but most often comes as condensation. In most cases, protecting from water is as easy as placing the CR200(X) in a weather tight enclosure with desiccant and elevating the enclosure above the ground. The CR200(X) is shipped with desiccant to reduce humidity. Desiccant should be changed periodically. Do not completely seal the enclosure if lead acid batteries are present; hydrogen gas generated by the batteries may build up to an explosive concentration.

### 3.1.8.2 Protection from Voltage Transients

---

**Read More!** See [Grounding](#) (p. 55).

---

The CR200(X) must be grounded to minimize the risk of damage by voltage transients associated with power surges and lightning induced transients. Earth grounding is required to form a complete circuit for voltage clamping devices internal to the CR200(X).

### 3.1.8.3 Calibration

---

**Read More!** See [Self-Calibration](#) p. 41.

---

The CR200(X) uses an internal voltage reference to routinely calibrate itself.

### 3.1.8.4 Replacing the Internal Battery

---

**Caution** Fire, explosion, and severe burn hazard! Misuse or improper installation of the lithium battery can cause severe injury. Do not recharge, disassemble, heat above 100°C (212°F), solder directly to the cell, incinerate, or expose contents to water. Dispose of spent lithium batteries properly.

---

The CR200(X) contains a lithium battery that operates the clock and SRAM when the CR200(X) is not powered. The CR200(X) does not draw power from the lithium battery while it is powered by an external source. In a CR200(X) stored at room temperature, the lithium battery should last approximately 5 years (less at temperature extremes). Where the CR200(X) is powered most or all of the time the lithium cell should last much longer.

The internal lithium battery must be replaced periodically. Factory replacement is recommended. Contact Campbell Scientific to obtain an RMA prior to shipping the CR200(X).

If the lithium cell is removed or allowed to discharge below the safe level, the CR200(X) will still operate correctly while powered. Without the lithium battery, the clock will reset and data is lost when power is removed.

A replacement lithium battery can be purchased from Campbell (part number 15598). [TABLE. CR200\(X\) Lithium Battery Specifications](#) p. 33 lists the specifications of the battery. However, Campbell Scientific recommends that the battery be replaced at the factory.

Table 3. Internal Lithium Battery Specifications	
Manufacturer	Renata
Model	CR2016 (3.6V)
Capacity	80 mAh
Self-discharge rate	1%/year @ 23°C
Operating temperature range	-40°C to +85°C

## 3.2 PC Support Software

---

**Read More!** See [Support Software](#) (p. 143).

---

Several datalogger support software products for Windows are available. Software for datalogger setup and simple applications, PC200W and Short Cut, are available at no cost at [www.campbellsci.com](http://www.campbellsci.com). For more complex programming, telecommunications, networking, and reporting features, full-featured products are available from Campbell Scientific.

- PC200W Starter Software is available at no charge at [www.campbellsci.com](http://www.campbellsci.com). It supports a transparent RS-232 connection between PC and CR200(X), and includes Short Cut for creating CR200(X) programs. Tools for setting the datalogger clock, sending programs, monitoring sensors, and on-site viewing and collection of data are also included.
- PC400 supports a variety of telecommunication options, manual data collection, and data monitoring displays. Short Cut and CRBASIC Editor are included for creating programs. PC400 does not support complex communication options, such as phone-to-RF, PakBus® routing, or scheduled data collection.
- LoggerNet supports combined telecommunication options, customized data monitoring displays, and scheduled data collection. It includes Short Cut and CRBASIC Editor for creating programs. It also includes tools for configuring, trouble-shooting, and managing datalogger networks. LoggerNet Admin and LoggerNet Remote are also available for more demanding applications.



## 3.3 Specifications

Electrical specifications are valid over a -40° to +50°C range unless otherwise specified; non-condensing environment required. We recommend that you confirm system configuration and critical specifications with Campbell Scientific before purchase.

### ANALOG INPUTS; DIGITAL I/O

Channels SE1 to SE5 can be individually configured for single-ended measurement or digital I/O.

#### SINGLE-ENDED MEASUREMENT (SE1 TO SE5):

Analog Input Range:  $0 \leq V < 2.5$  Vdc  
 Input should not exceed 4 Vdc  
 Measurement Resolution: 0.6 mV  
 Measurement Accuracy  
   Typical:  $\pm(0.25\%$  of reading + 1.2 mV offset)  
   over -40° to +50°C  
   Worst-case:  $\pm(1\%$  of reading + 2.4 mV offset)  
   over -40° to 50°C

#### DIGITAL I/O (SE1 TO SE5):

Input/Output High State: 2.1 to 3.3 Vdc  
 Input/Output Low State: <0.9 Vdc  
 Output High State: 3.3 V (no load)  
 Drive Current: 220  $\mu$ A @ 2.7 Vdc  
 Maximum Input Voltage: 4 Vdc

#### HALF BRIDGE MEASUREMENTS:

Accuracy: Relative to the excitation.  
 Using +2.5 Vdc excitation, is  
 $\pm(0.06\%$  of reading + 2.4 mV)/(2.5 Vdc)

#### PERIOD AVERAGING (SE1 TO SE4):

Maximum Input Voltage: 4 Vdc  
 Frequency Range: 0 to 150 kHz  
 Voltage Threshold: counts cycles on transition  
 from <0.9 Vdc to >2.1 Vdc

#### EXCITATION CHANNELS (VX1 AND VX2):

Range: Programmable 0, 2.5, 5 Vdc, or  
 off (floating)  
 Accuracy:  $\pm 25$  mV on +2.5 Vdc range,  $\pm 125$  mV  
 on +5.0 Vdc range  
 Maximum Current: 25 mA on +2.5 Vdc range,  
 10 mA on +5.0 Vdc range

### CONTROL PORTS (C1 AND C2)

#### DIGITAL I/O:

Voltage Level When Configured as Input:  
 <0.9 Vdc (low state) to >2.7 Vdc (high state)  
 Voltage Level When Configured as Output:  
 0 V (low state), 5 Vdc (high state) (no load)  
 Logic Level: TTL  
 Drive Current: 1.5 mA @ 4.5 V

SDI-12: SDI-12 sensors connect to C1

### PULSE COUNTERS

#### SWITCH CLOSURE (P\_SW):

Maximum Count Rate: 100 Hz  
 Minimum Switch Open Time: 5 ms  
 Minimum Switch Closed Time: 5 ms  
 Maximum Bounce Time: 4 ms

#### PULSE COUNT (P\_SW, C1, AND C2):

Voltage Threshold: count on transition from  
 <0.9 V to >2.7 Vdc  
 Minimum Pulse Width: 320  $\mu$ s  
 Maximum Input Frequency: 1 kHz  
 Max Input Voltage: C1 & C2 ( $\pm 12$  Vdc), P\_SW (4 Vdc)  
 P\_LL ( $\pm 20$  Vdc)

#### LOW LEVEL AC (P\_LL):

Voltage Threshold: <0.5 to >2 V  
 Minimum Input: 20 mV RMS

Maximum Frequency: 1 kHz

Note: C1 and C2 can be used for switch closure  
 using the battery voltage and a 100 kOhm  
 pull-up resistor. If the dc offset is >0.5 V, then  
 AC coupling is required.

### COMMUNICATIONS

SERIAL INTERFACE: Female RS-232 9-pin  
 interface for logger-to-PC communications

#### ON-BOARD SPREAD SPECTRUM RADIO:

Frequency: 915 MHz (CR206X),  
 922 MHz (CR211X), or 2.4 GHz (CR216X)  
 Transmission Range: 1 mile with 0 dBd  $\frac{1}{4}$  wave  
 antenna (line-of-sight) and 900 MHz radios;  
 0.6 miles (1 km) with 0 dBd  $\frac{1}{2}$  wave antenna  
 (line-of-sight) and 2.4 GHz radio;  
 up to 10 miles with higher gain antenna  
 (line-of-sight)  
 RF4XX used as a base station radio

#### AVAILABLE RADIO TRANSMISSION MODES:

Always on, program controlled  
 Cycle Time: 1 or 8 s cycles; on for 100 ms every  
 period; checks for incoming communication  
 Scheduled Transmission Time: off until transmis-  
 sion time  
 PakBus<sup>®</sup> packet switching network protocol

### CLOCK ACCURACY

8.2 minutes/month @ -40° to +50°C; 1 minute/month  
 @ +25°C

### CPU AND STORAGE

FINAL STORAGE: 500 kbyte Flash, data format  
 is 4 bytes per data point (table-based)

SRAM: 8 kbytes

COMPILED PROGRAM STORAGE: up to 19.6 kbyte  
 depending on structure of CRBasic program

OPERATING SYSTEM FLASH: 106 kbyte

FASTEST SCAN RATE: once per second

### SWITCHED BATTERY (SW BATTERY)

Switched under program control; 300 mA minimum  
 current available

### POWER

BATTERY VOLTAGE RANGE: 7 to 16 Vdc  
 (can program datalogger to measure internal  
 battery voltage)

MAXIMUM CONTINUOUS BATTERY CHARGING  
 CURRENT:  
 0.9 A @ 20°C; 0.65 A @ 50°C

RECOMMENDED BATTERIES: 12 Vdc, 7 Ah or  
 smaller sealed rechargeable battery when  
 connected to the on-board charging circuit.  
 Using larger batteries with the datalogger's  
 built-in charger may result in excessive PC  
 board heating. This is especially a concern  
 when the battery is deeply discharged or failing  
 with a shorted cell.

Alkaline cells, lithium, or other non-rechargeable  
 battery types may be connected if the charging  
 circuit is not used (i.e., nothing connected to  
 Charge terminals).

CHARGER INPUT VOLTAGE: 16 to 22 Vdc

SOLAR PANEL: 10 W or smaller when using  
 on-board charging circuit.

WALL CHARGER: 1 A or smaller when using on-  
 board charging circuit.

SHELF LIFE OF CLOCK'S BACKUP BATTERY:  
 5 years

### CURRENT DRAIN (@12 V)

QUIESCENT CURRENT DRAIN:  
 No Radio or Radio Powered Off: ~0.2 mA

#### ACTIVE CURRENT DRAIN:

No radio ~3 mA  
 Radio receive ~20 mA (CR206X, CR211X),  
 ~36 mA (CR216X)  
 Radio transmit ~75 mA (CR206X, CR211X,  
 CR216X)

#### AVERAGE CONTINUOUS CURRENT DRAIN:

Radio always on ~20 mA (CR206X, CR211X),  
 ~36 mA (CR216X)  
 Radio in 1 s duty cycle ~2.2 mA (CR206X,  
 CR211X), ~4 mA (CR216X)  
 Radio in 8 s duty cycle ~0.45 mA (CR206X,  
 CR211X), ~0.8 mA (CR216X)

### CE COMPLIANCE (as of 03/02)

CE COMPLIANT DATALOGGERS: CR200X, CR206X,  
 CR211X. CR216 and CR216X are not CE compliant.

STANDARD(S) TO WHICH CONFORMITY IS  
 DECLARED: IEC61326:2002

### EMI AND ESD PROTECTION

IMMUNITY: Meets or exceeds following standards:

ESD: per IEC 1000-4-2;  $\pm 8$  kV air,  $\pm 4$  kV contact  
 discharge  
 RF: per IEC 1000-4-3; 3 V/m, 80-1000 MHz  
 EFT: per IEC 1000-4-4; 1 kV power, 500 V I/O  
 Surge: per IEC 1000-4-5; 1 kV power and I/O  
 Conducted: per IEC 1000-4-6; 3 V 150 kHz-80 MHz

Emissions and immunity performance criteria avail-  
 able on request.

### PHYSICAL

CASE DESCRIPTION: Aluminum with spring-loaded  
 terminals

DIMENSIONS (including terminals):  
 5.5 in x 3 in x 2 in; 14.0 cm x 17.6 cm x 5.1 cm

#### WEIGHT:

CR200X or CR295X: 8.5 oz (242 g)  
 CR206X, CR211X, or CR216X: 9.5 oz (271 g)

CUSTOM CASE: available for OEM applications;  
 contact Campbell Scientific

### WARRANTY

Three year covering parts and labor.



## Section 4. Sensor Support

---

Several features give the CR200(X) the flexibility to measure many sensor types. Contact a Campbell Scientific applications engineer if assistance is required to assess sensor compatibility.

### 4.1 Powering Sensors

---

**Read More!** See [CR200\(X\) Power Supply](#) (p. 53).

---

The CR200(X) is a convenient router of power for sensors and peripherals requiring a 2.5, 5 or 12 VDC source. It has one program-controlled switched 12 Volt terminal (SW Battery), and two switched voltage excitation terminals (VX1, VX2). These terminals limit current internally for protection against accidental short circuits. Sensors can be powered continuously by connecting to the Battery + terminal, which is the same as connecting sensors directly to the external battery.

Voltage on the Battery + and SW Battery terminals will change with the DC supply used to power the CR200(X). The switched voltage excitation terminals are typically used to supply a brief excitation for a bridge measurement with the ExDelSE () instruction, but can also be programmed to provide a continuous 2.5 V or 5 V using the ExciteV () instruction, which is enough to power many sensors. [Switched Unregulated \(Nominal 12 Volt\)](#) (p. 38) shows the current limits of SW Battery. Greatly reduced output voltages associated with SW Battery, VX1, and VX2 due to current limit may occur if the current limits given in the table in [TABLE. Current Sourcing Limits](#) p. 38 are exceeded.

#### 4.1.1 Switched Precision

**Voltage (0, 2500, 5000 mV)**

Two switched analog output (excitation) terminals, VX1 and VX2, operate under program control. Check the accuracy specification of these channels in [Specifications](#) (p. 35) to understand their limitations. Specifications are only applicable for loads not exceeding  $\pm 25$  mA for 2.5 V excitation, or 10 mA for 5 V excitation. CRBASIC instructions that control excitation channels include:

- ExDelSE ()
- ExciteV ()

#### 4.1.2 Continuous Unregulated (Nominal 12 Volt)

Voltage on the Battery + terminal will change with the CR200(X) external battery voltage.

### 4.1.3 Switched Unregulated (Nominal 12 Volt)

Voltage on the SW Battery terminal will change with CR200(X) supply voltage. The CRBASIC instruction SWBatt () controls the SW Battery terminal.

Table 4. Current Sourcing Limits	
Terminal	Limit
VX1, VX2	25 mA @ 2.5V 10 mA @ 5V
SW Battery	< 900 mA @ 20°C < 729 mA @ 40°C < 630 mA @ 50°C < 567 mA @ 60°C < 400 mA @ 85°C

## 4.2 Voltage Measurement

### 4.2.1 Measurement Sequence

The first step in a voltage measurement is a calibration to measure the ground offset. The calibration is performed once for each voltage measurement. The CR200(X) measures analog voltages with a sample and hold analog to digital (A/D) conversion. The A/D conversion is made with a 12-bit successive approximation technique which resolves the signal voltage to one part in 4096 of the full scale 2.5 V range, which is 0.6 millivolts.

To reduce noise, 10 measurements are rapidly made and averaged to form the result returned. The measurements that go into the average each take about 26 microseconds.

### 4.2.2 Measurement Accuracy

CR200(X) analog measurement error is calculated as

$$\text{Error} = \text{Gain Error (\%)} + \text{Offset Error}$$

Gain error is expressed as  $\pm\%$  and is a function of input voltage and CR200(X) temperature. It increases with component temperature and aging. Between -40°C and +50°C, gain error is typically  $\pm 0.25\%$  of the reading with a 1.2 millivolt offset. Worst case over that same temperature range is  $\pm 1\%$  of the reading with a 2.4 millivolt offset.

*FIGURE. Voltage Measurement Accuracy (-40° to +50°C)* p. 40 illustrates that as magnitude of input voltage decreases, measurement error decreases.

---

**Note** The accuracy specification includes only the CR200(X)'s contribution to measurement error. It does not include the error of sensors.

---

For example, assume the following (see [Specifications](#) (p. 35)):

- Input Voltage: +2000 mV
- Programmed Measurement Instruction: VoltSE ()
- CR200(X) Temperature: Between -40°C and +50°C

Accuracy of the measurement is calculated as follows:

$$\text{Error} = \text{Gain Error} + \text{Offset Error},$$

where

$$\begin{aligned}\text{Gain Error} &= \pm (2000 * 0.0025) \\ &= \pm 5 \text{ mV}\end{aligned}$$

and

$$\text{Offset Error} = 1.2 \text{ mV}$$

Therefore,

$$\begin{aligned}\text{Error} &= \text{Gain Error} + \text{Offset Error} \\ &= \pm 5 \text{ mV} + 1.2 \text{ mV} \\ &= \pm 6.2 \text{ mV}\end{aligned}$$

A worst case accuracy in these conditions is

$$\text{Error} = \pm(2000 \text{ mV} * 0.01 + 2.4 \text{ mV}) = \pm 22.4 \text{ mV}$$

In contrast, the error for a 500 mV input under the same constraints is  $\pm 2.45 \text{ mV}$  typical and  $\pm 7.4 \text{ mV}$  worst case.

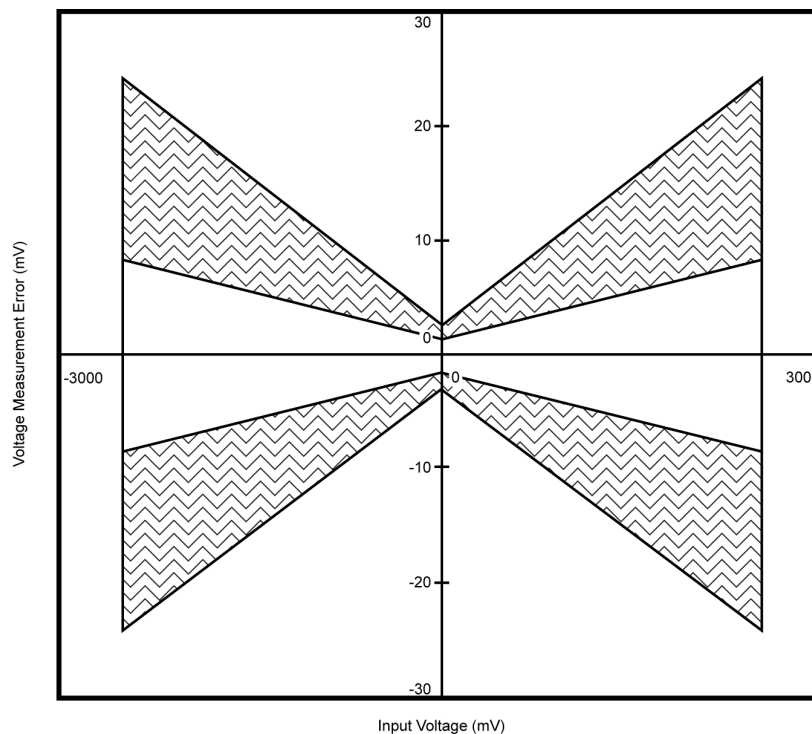


Figure 27: Voltage Measurement Accuracy (0° to 40° C)

### 4.2.3 Voltage Range

The CR200(X) has one analog voltage range of 0 to 2.5 volts. The resolution for a single A/D conversion is 0.6 millivolts.

### 4.2.4 Integration

Integration is used to reduce the noise included in a measurement. The CR200(X) uses a form of digital integration. It makes 10 A/D conversions and averages them for the result returned. The A/D conversions are made every 26 microseconds.

Averaging will also reduce signal noise from the sensor such as when water level measured with a pressure transducer is changing because of pressure fluctuations caused by wind.

Averaging also has an effect on the resolution. The resolution seen in the numerical result is the resolution of a single A/D conversion (0.6 mV) divided by the number of A/D conversions (10).

4.2.5 Self-Calibration

A calibration measurement to measure the ground offset is made at the beginning of each measurement instruction that includes a voltage measurement. This calibration takes about 400 microseconds. Only one calibration measurement is made per instruction regardless of the number of reps.

The battery voltage is checked every 8 seconds to ensure it is within the allowable range.

4.3 Bridge Resistance Measurements

Many sensors detect phenomena by way of change in a resistive circuit. Thermistors, strain gages, and position potentiometers are examples. Resistance measurements are special case voltage measurements. By supplying a precise, known voltage to a resistive circuit, then measuring the returning voltage, resistance can be calculated.

Two bridge measurement instructions are included in the CR200(X), ExDelSE () and Therm109 (). ExDelSE () is used with sensors that have a simple half bridge circuit. Therm109 () is used with Campbell Scientific’s 109-L thermistor probe. Sensors with bridge circuits that require a differential voltage measurement, such as full bridge or 3-wire half bridge, cannot be measured with the CR200(X).

*FIGURE. Half Bridge Circuit Used with ExDelSE* (p. 41) shows the circuit that is typically measured with ExDelSE (). In the diagram, Rs is normally the sensor and Rf is normally a precision fixed (static) resistor. Vx is the excitation voltage (either 2500 or 5000 mV) and V1 is the voltage (mV) measured by the analog input channel.

Calculating the resistance of a sensor that is one of the legs of a resistive bridge requires additional processing following the bridge measurement instruction. *FIGURE. Half Bridge Circuit Used with ExDelSE* (p. 41) lists the schematics of a typical half bridge configuration and the calculations necessary to compute the resistance of any single resistor, provided the value of the other resistor in the bridge circuit is known.

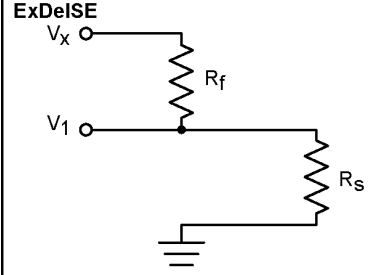
Sensor Schematic	Base Equation	Formulae
<div><p><b>ExDelSE</b></p></div>	<div><p>Result with Mult = 1/ExmV Offset = 0</p><math display="block">X = \frac{V_1}{V_x} = \frac{R_s}{R_s + R_f}</math></div>	<div><math display="block">R_s = R_f \frac{X}{1-X}</math><math display="block">R_f = \frac{R_s(1-X)}{X}</math></div>

Figure 28: Voltage Excitation Bridge Circuit

### 4.3.1 Measurements Requiring AC Excitation

Some resistive sensors require AC Excitation. These include electrolytic tilt sensors, soil moisture blocks, water conductivity sensors, and wetness sensing grids. The use of DC excitation in these sensors can result in polarization, which will cause erroneous measurement, shift calibration, or lead to rapid sensor decay.

Other sensors, e.g., LVDTs (Linear Variable Differential Transformer), require and AC excitation because they rely on inductive coupling to provide a signal. DC excitation will provide no output.

CR200(X) bridge measurements cannot reverse excitation polarity to provide AC excitation and avoid ion polarization. Sensors requiring AC excitation should not be used with the CR200(X).

Other Campbell Scientific dataloggers (e.g. CR800 series, CR1000, CR3000) are compatible with sensors that require AC excitation.

## 4.4 Pulse Count Measurement

*FIGURE. Switch Closure Pulse Sensor* p. 42 is a generalized schematic showing connection of a pulse sensor to the CR200(X). The CR200(X) features two dedicated pulse input channels, P\_SW and P\_LL, and two digital I/O channels, C1 and C2, for measuring pulse output sensors. Activated by the PulseCount () instruction, dedicated 16-bit counters on P\_SW, P\_LL, C1 and C2 are used to accumulate all counts over the user specified scan interval. The value which is output for each scan is the difference in the last known counter value and the new counter value. Since the last count is maintained for each scan, even if the counter rolls over between scans the correct count will be recorded. If the time between scans is such that the counter exceeds 65,536 pulses during a scan, then the counter will roll over twice resulting in an erroneous measurement. PulseCount () instruction parameters specify the pulse input type, channel used, and pulse output option.

---

Note: The PulseCount instruction must be executed once before the pulse or control port is ready for input. This may be of particular concern for programs with long scan intervals. For example, the PulseCount () instruction will not yield a valid output until the turn of the second hour if the PulseCount () instruction is used within a program with a scan interval of 1 hour.

---

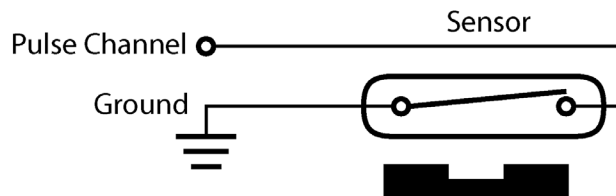


Figure 29: Switch Closure Pulse Sensor



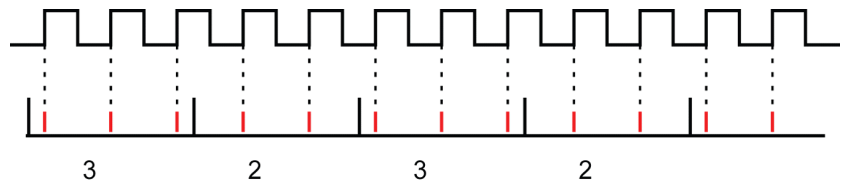
---

**Note:** The PulseCount () instruction should not be used in a conditional statement or subroutine. To ensure all pulses are detected, it must be executed each scan.

---

Execution of PulseCount () within a scan involves determining the accumulated counts in each dedicated 16-bit counter since execution of the last PulseCount (). PulseCount () parameter (POption) determines if the output is in counts (POption = 0) or frequency (POption = 1). Counts are the preferred output option for measuring number of tips from a tipping bucket rain gage, or the number of times a door opens. Many pulse sensors, such as anemometers and flow meters, are calibrated in terms of frequency (Hz or counts / second), and are usually measured with the frequency option.

Resolution of the pulse counters is one count. Resolution of frequency is (1/scan interval). For example, the frequency resolution of PulseCount() returning a result every 1 second is 1 Hz. The resultant measurement will bounce around by the resolution. For example, if you are scanning a 2.5 Hz input once a second, in some intervals there will be 2 counts and in some 3. If the pulse measurement is averaged, the correct value will be the result.



Accuracy is limited by a small scan interval error of  $\pm(3 \text{ ppm of scan interval} + 10 \text{ } \mu\text{s})$  plus the measurement resolution error of  $\pm 1 \text{ Hz}$ . The sum is essentially  $\pm 1 \text{ Hz}$ . Extending a 1 second measurement interval to 10 seconds, either by increasing the scan interval or by averaging, improves the resulting frequency resolution from 1 Hz to 0.1 Hz. Averaging can be accomplished by the Average () instruction or by computing a running or spatial average through programming.

#### 4.4.1 Pulse input Channels

---

Read More! Review [pulse counter specifications](#) p. 35. Review pulse counter programming in CRBASIC Help for the PulseCount () instruction.

---

**FIGURE. Pulse Input Types** (p. 7) illustrates pulse input types measured by the CR200(X). Dedicated pulse input channel P\_SW can be configured to read high-frequency pulses or switch closure, while P\_LL can be configured to read a low-level AC signal. With a 100 kOhm pull-up resistor added to the wiring panel P\_LL, C1, or C2 can measure pulse input signals (See [Pulse Input \(P\\_SW\)](#) p. 44).

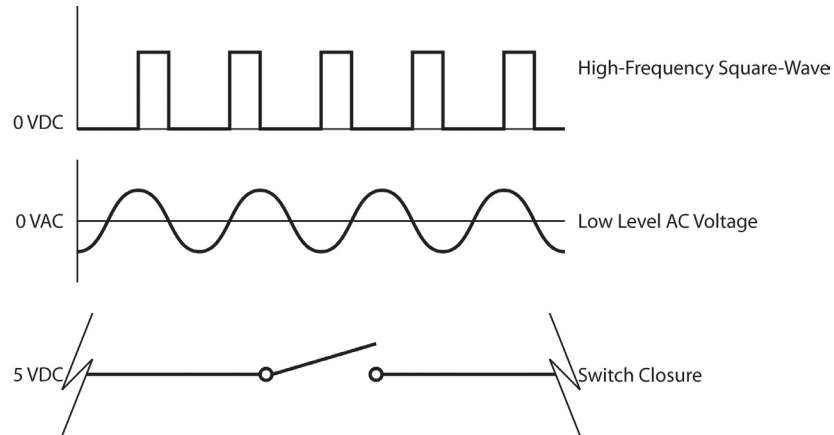


Figure 30: Pulse Input Types

#### 4.4.1.1 Pulse Input (P\_SW)

Internal hardware routes high-frequency pulse to an inverting CMOS input buffer with input hysteresis. The CMOS input buffer is guaranteed to be an output zero level with its input  $\geq 2.7$  V, and guaranteed to be an output one with its input  $\leq 0.9$  V. An RC input filter with approximately a 1  $\mu$ s time constant precedes the inverting CMOS input buffer, resulting in an amplitude reduction of high frequency signals between the P\_SW terminal block and the inverting CMOS input buffer as illustrated in FIGURE. Amplitude reduction of pulse-count waveform. For a 0 to 5 Volt square wave applied to P\_SW, the maximum frequency that can be counted in pulse input mode is approximately 1 kHz.

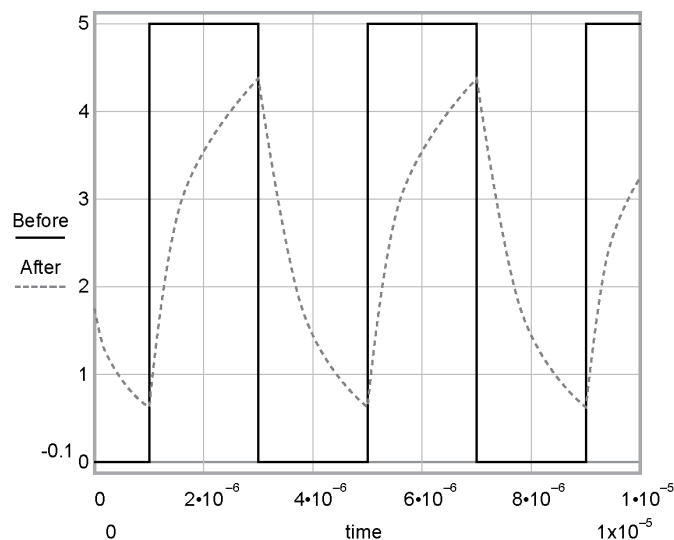


Figure 31: Amplitude Reduction of Pulse-Count Waveform (before and after 1 ms time constant filter)

When a pulse channel is configured for pulse input mode, an internal 100 k $\Omega$  pull-up resistor to +5 Volt on the P\_SW input is automatically employed. This pull-up resistor accommodates open-collector (open-drain) output devices for high-frequency input. An external 100 k $\Omega$  pull-up resistor connecting Battery+ to P\_LL, C1, or C2 must be added to perform pulse counting on any of those channels (See [Multiple Switch Closure Measurements](#) p. 112).

The maximum input voltage on P\_SW is 4 volts. For C1 and C2 the maximum input voltage is 6.5 volts.

#### 4.4.1.2 Low-Level AC (P\_LL)

Rotating magnetic pickup sensors commonly generate AC output voltages ranging from millivolts at low rotational speeds to several volts at high rotational speeds. Channel P\_LL contains internal signal conditioning hardware for measuring low-level AC output sensors. P\_LL measure signals ranging from 20 mV RMS ( $\pm 28$  mV peak) to 14 V RMS ( $\pm 20$  V peak). Internal AC coupling is incorporated in the low-level AC hardware to eliminate DC offset voltages of up to  $\pm 0.5$  V.

#### 4.4.2 Pulse Input on Digital I/O Channels C1–C2

---

Read More! Review digital I/O channel specifications in [Specifications](#) p. 35. Review pulse counter programming in CRBASIC Help for the PulseCount () instruction.

---

Digital I/O channels C1 – C2 can be configured to measure pulse input signals. Pulse input signals require external 100 k $\Omega$  pull-up resistors to be connected between Battery+ and the control ports. Maximum input voltage level is 6.5 V. If pulse inputs greater than +6.5 V need to be measured, external signal conditioning must employed. Contact Campbell Scientific for further information.

Low-level AC signals cannot be measured directly on channels C1 – C2.

### 4.5 Period Averaging Measurements

The CR200(X) can measure the period of a signal on any single-ended analog input channel (SE 1 - 5). The specified number of cycles are timed with a resolution of 70 ns, making the resolution of the period measurement 70 ns divided by the number of cycles chosen.

Cycles are counted as the input voltage transitions from less than 0.9 volts to more than 2.1 volts. The maximum input voltage must be less than 4 volts. The maximum frequency that can be detected is 150 kHz.

## 4.6 SDI-12 Recording

---

**Read More!** [SDI-12 Sensor Support](#) p. 112 and [Serial Input / Output](#) p. 105.

---

SDI-12 is a communications protocol developed to transmit digital data from smart sensors to data acquisition units. It is a simple protocol, requiring only a single communication wire. Typically, the data acquisition unit also supplies power (12V and ground) to the SDI-12 sensor. The CR200(X) is equipped with 1 SDI-12 channel and an SDI12Recorder () CRBASIC instruction.

## 4.7 Cabling Effects on Measurements

Sensor cabling can have significant effects on sensor response and accuracy. This is usually only a concern with sensors acquired from manufacturers other than Campbell Scientific. Campbell Scientific sensors are engineered for optimal performance with factory installed cables.

### 4.7.1 Analog Sensor Cables

Cable length in analog sensors is most likely to affect the signal settling time. For more information, see Signal Settling Time.

### 4.7.2 Pulse Sensors

Because of the long interval between switch closures in tipping bucket rain gages, appreciable capacitance can build up between wires in long cables. A built up charge can cause arcing when the switch closes, shortening switch life. As shown in [FIGURE. Current Limiting Resistor in a Rain Gage Circuit](#) (p. 46), a 100 ohm resistor is connected in series at the switch to prevent arcing. This resistor is installed on all rain gages currently sold by Campbell Scientific.

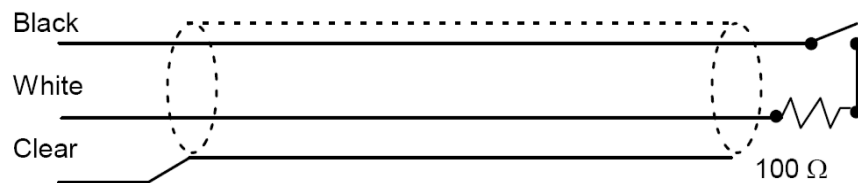


Figure 32: Current Limiting Resistor in a Tipping Bucket Rain Gage Circuit

### **4.7.3 Serial Sensors**

#### **4.7.3.1 SDI-12 Sensors**

The SDI-12 standard allows cable lengths of up to 200 feet. Campbell Scientific does not recommend SDI-12 sensor lead lengths greater than 200 feet; however, longer lead lengths can sometimes be accommodated by increasing the wire gage and/or powering the sensor with a second 12 vdc power supply placed near the sensor.



# Section 5. Measurement and Control Peripherals

---

Peripheral devices expand the CR200(X) input / output capacity. Classes of peripherals are discussed below according to use.

---

**Read More!** For complete information on available measurement and control peripherals, go to [APPENDIX. Sensors and Peripherals](#) (Appendix p. 28), [www.campbellsci.com](http://www.campbellsci.com), or contact a Campbell Scientific applications engineer.

---

## 5.1 Control Output

Controlling power to an external device is a common function of the CR200(X). On-board control terminals are available for binary (on / off) control.

Many devices are conveniently controlled with the SW Battery (Switched 12 Volt) terminal on the CR200(X). Applications requiring more control channels or greater power sourcing capacity may be satisfied by using control ports C1 – C2 in conjunction with single-channel switching relays.

### 5.1.1 Binary Control

#### 5.1.1.1 Digital I/O Ports

Each of 2 digital I/O ports (C1 - C2) can be configured as an output port and set low (0 V) or high (5 V) using the PortSet () or WriteIO () instructions. A digital I/O port is normally used to operate an external relay driver circuit because the port itself has limited drive capacity. Drive capacity is determined by the 5V supply and a 330 ohm output resistance. It is expressed as:

$$V_o = 4.9V - (330 \text{ ohms}) * I_o$$

Where  $V_o$  is the drive limit, and  $I_o$  is the current required by the external device. [FIGURE. Control Port Current Sourcing](#) (p. 50) plots the relationship.

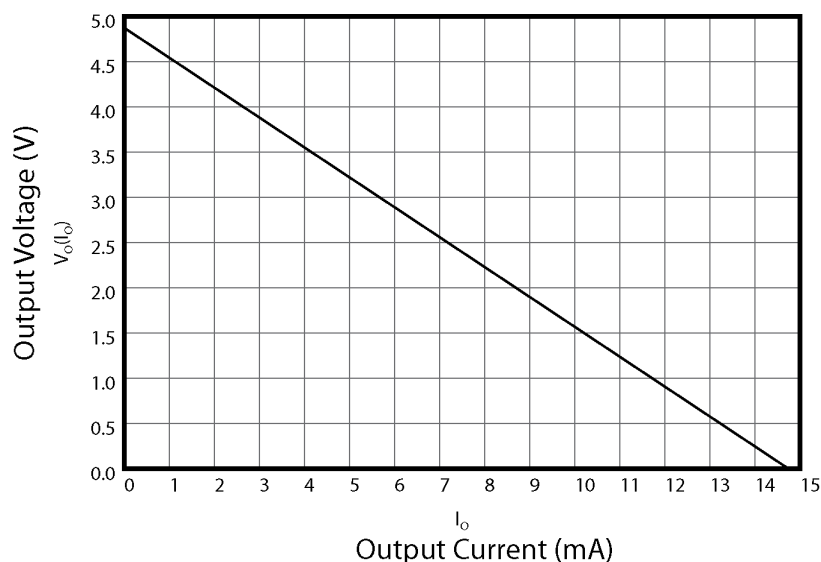


Figure 33: Control Port Current Sourcing

#### 5.1.1.2 Switched 12 V Control

The SW Battery port can be set low (0 V) or high (12 V) using the SWBatt () instruction. The port is often used to control low power devices such as sensors that require 12 V during measurement. Current sourcing must be limited to 900 mA or less at 20°C. See [TABLE. Current Sourcing Limits](#) (p. 38).

A 12V switching circuit, driven by a digital I/O port, is also available from Campbell Scientific.

---

**Note** The SW Battery supply is unregulated and can supply up to 900 mA at 20°C. See [TABLE. Current Sourcing Limits](#) (p. 38). A resettable polymeric fuse protects against over-current. Reset is accomplished by removing the load or turning off the SW Battery for several seconds.

---

#### 5.1.1.3 Relays and Relay Drivers

Compatible, inexpensive and reliable single-channel relay drivers for a wide range of loads are available from various electronic vendors such as Crydom, Newark, Mouser, etc.

#### 5.1.1.4 Component Built Relays

[FIGURE. Relay Driver Circuit with Relay](#) (p. 51) shows a typical relay driver circuit in conjunction with a coil driven relay which may be used to switch external power to some device. In this example, when the control port is set high, 12 V from the datalogger passes through the relay coil, closing the relay which completes the power circuit and turns on the fan.



In other applications it may be desirable to simply switch power to a device without going through a relay. [FIGURE. Power Switching without Relay](#) (p. 51) illustrates a circuit for switching external power to a device without using a relay. If the peripheral to be powered draws in excess of 75 mA at room temperature (limit of the 2N2907A medium power transistor), the use of a relay is required.

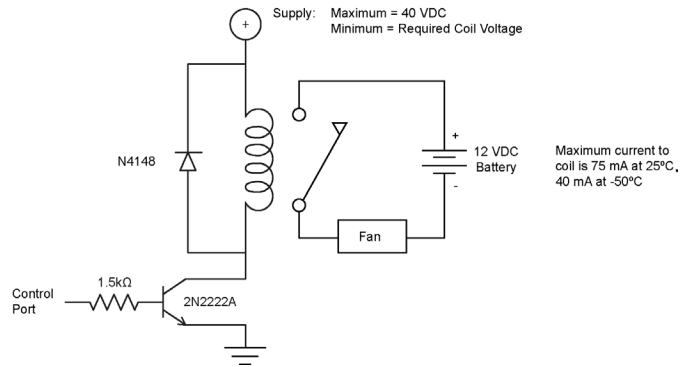


Figure 34: Relay Driver Circuit with Relay

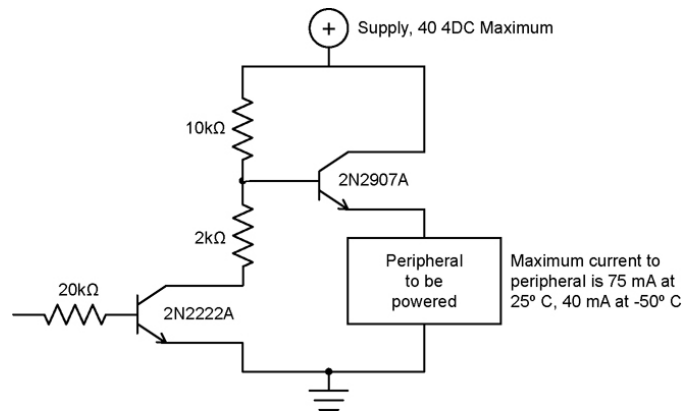


Figure 35: Power Switching without Relay

## 5.2 Other Peripherals

### 5.2.1 TIMs

Terminal Input Modules (TIMs) are devices that provide simple measurement support circuits in a convenient package. TIMs include voltage dividers for cutting the output voltage of sensors to voltage levels compatible with the CR200(X), modules for completion of resistive bridges, and shunt modules for measurement of analog current sensors.



## Section 6. CR200(X) Power Supply

---

**Reliable power is the foundation of a reliable data acquisition system.** When designing a power supply, consideration should be made regarding worst-case power requirements and environmental extremes.

Contact Campbell Scientific if assistance in selecting a power supply is needed, particularly with applications in extreme environments.

### 6.1 Power Requirement

The CR200(X) operates from a DC power supply with voltage ranging from 7 to 16 V. It is internally protected against accidental polarity reversal. Input voltages in excess of 18 V may damage the CR200(X) and/or power supply.

---

**Caution** The Battery + and Battery - terminals on the wiring panel are not regulated by the CR200(X); they obtain the same power directly the POWER IN terminal. When using the CR200(X) wiring panel to source power to other 12 V devices, be sure the power supply regulates the voltage within a the range specified by the manufacturer of the connected device.

---

### 6.2 Calculating Power Consumption

---

**Read More! Power Requirements** (p. 29).

---

System operating time for batteries can be determined by dividing the battery capacity (ampere-hours) by the average system current drain (amperes). The CR200(X) typically has an average current drain of less than 3 mA. The duty cycle of the radio will add to the current drain. Be sure to include the current used by any powered sensors when calculating power supply requirements.

### 6.3 Power Supplies

Complete power supply information is available in manual or brochure form at [www.campbellsci.com](http://www.campbellsci.com).

#### 6.3.1 Battery Connection

Any clean, 7 to 16 VDC supply may be connected to the Battery + and Battery - connector terminals on the front panel. When connecting external power to the CR200(X), insert the positive lead into the Battery + terminal. Insert the ground lead into the Battery - terminal.

Auxiliary photovoltaic power sources may be used to maintain charge on lead acid batteries. Unregulated solar panels may be connected to Charge + and Charge – channels on the datalogger wiring panel.

When selecting a solar panel, a rule-of-thumb is that on a stormy overcast day the panel should provide enough charge to meet the system current drain (assume 10% of average annual global radiation, kW/m<sup>2</sup>). Specific site information, if available, could strongly influence the solar panel selection. For example, local effects such as mountain shadows, fog from valley inversion, snow, ice, leaves, birds, etc. shading the panel should be considered.

If help is needed in determining system power requirements, contact Campbell Scientific's Marketing Department.

# Section 7. Grounding

---

Grounding the CR200(X) and its peripheral devices and sensors is critical in all applications. Proper grounding will ensure the maximum ESD (electrostatic discharge) protection and higher measurement accuracy.

## 7.1 ESD Protection

ESD (electrostatic discharge) can originate from several sources, the most common, and most destructive, being primary and secondary lightning strikes. Primary lightning strikes hit the datalogger or sensors directly. Secondary strikes induce a voltage in power lines or sensor wires.

The primary devices for protection against ESD are gas-discharge tubes (GDT). All critical inputs and outputs on the CR200(X) are protected with GDTs or transient voltage suppression diodes. GDTs fire at 150 V to allow current to be diverted to the earth ground lug. To be effective, the earth ground lug must be properly connected to earth (chassis) ground. The power ground and signal grounds have independent paths to the ground lug.

The 9-pin serial port is another path for transients. Communications paths such as a telephone or short-haul modem lines should be provided spark gap protection at installation. Spark gap protection is often an option with these products, so it should always be requested when ordering. Spark gaps for these devices must be connected to either the earth ground lug, the enclosure ground, or to the earth (chassis) ground.

A good earth (chassis) ground will minimize damage to the datalogger and sensors by providing a low resistance path around the system to a point of low potential. Campbell Scientific recommends that all dataloggers be earth (chassis) grounded. All components of the system (dataloggers, sensors, external power supplies, mounts, housings, etc.) should be referenced to one common earth (chassis) ground.

In the field, at a minimum, a proper earth ground will consist of a 6 to 8 foot copper sheathed grounding rod driven into the earth and connected to the CR200(X) Ground Lug with a 12 AWG wire. In low conductive substrates, such as sand, very dry soil, ice, or rock, a single ground rod will probably not provide an adequate earth ground. For these situations, consult the literature on lightning protection or contact a qualified lightning protection consultant.

In vehicle applications, the earth ground lug should be firmly attached to the vehicle chassis with 12 AWG wire or larger.

In laboratory applications, locating a stable earth ground is challenging, but still necessary. In older buildings, new AC receptacles on older AC wiring may indicate that a safety ground exists when in fact the socket is not grounded. If a safety ground does exist, it is good practice to verify that it carries no current. If the integrity of the AC power ground is in doubt, also ground the system through the buildings, plumbing or another connection to earth ground.

### 7.1.1 Lightning Protection

The most common and destructive ESDs are primary and secondary lightning strikes. Primary lightning strikes hit instrumentation directly. Secondary strikes induce voltage in power lines or wires connected to instrumentation. While elaborate, expensive and nearly infallible lightning protection systems are available, Campbell Scientific has for many years employed a simple and inexpensive design that protects most systems in most circumstances. It is, however, not infallible.

---

**Note** Lightning strikes may damage or destroy the CR200(X) and associated sensors and power supplies.

---

In addition to protections discussed in [ESD Protection](#) (p. 55), use of a simple lightning rod and low-resistance path to earth ground is adequate protection in many installations. A lightning rod serves two purposes. Primarily, it serves as a preferred strike point. Secondly, it dissipates charge, reducing the chance of a lightning strike. [FIGURE. Lightning Protection Scheme](#) (p. 57) shows a simple lightning protection scheme utilizing a lightning rod, metal mast, heavy gage ground wire, and ground rod to direct damaging current away from the CR200(X).

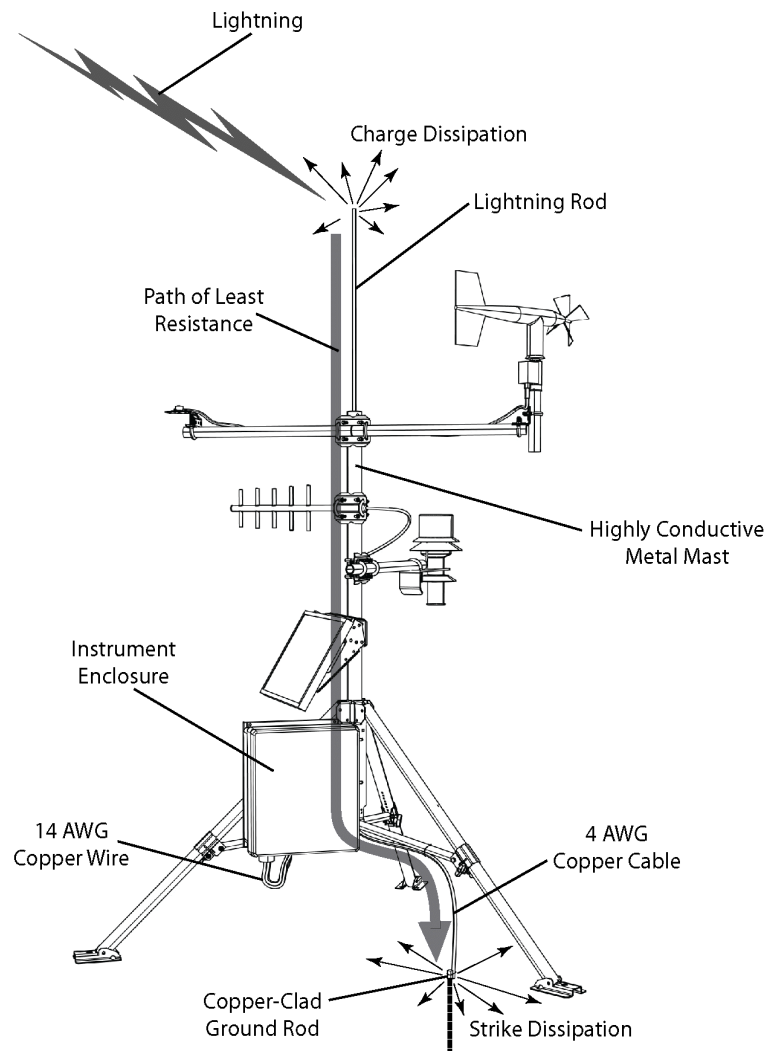


Figure 36: Lightning Protection Scheme

## 7.2 Single-Ended Measurement Reference

Low-level single-ended voltage measurements are sensitive to ground potential fluctuations. Although the CR200(X) is not sensitive enough for low-level measurements and ground potential fluctuations are not usually a problem, the grounding scheme in the CR200(X) has been designed to eliminate ground potential fluctuations due to changing return currents from SW Battery, excitation channels, and the control ports. This is accomplished by utilizing separate signal and shield grounds ( $\square$ ) and power grounds (G). To take advantage of this design, observe the following grounding rule:

**Note** Always connect a device's ground next to the active terminal associated with that ground. Several ground wires can be connected to the same ground terminal.

Examples:

- Connect grounds associated with SW Battery, VX1 (EX1), VX2 (EX2), C1, and C2 to G terminals.
- Connect the low side of single-ended sensors to the nearest (□) terminal on the analog input terminal blocks.
- Connect shield wires to the nearest (□) terminal on the analog input terminal blocks.



## Section 8. CR200(X) Configuration

---

The CR200(X) may require changes to factory default settings depending on the application. Most settings concern telecommunications between the CR200(X) and a network or PC.

---

**Note** The CR200(X) is shipped factory ready with all settings and firmware necessary to communicate with a PC via RS-232 and to accept and execute user application programs.

---

Prior to running DevConfig, connect a serial cable from the computer COM port to the RS-232 on the datalogger as shown in *Figure. Power and RS-232 Connections* p. 11.

### 8.1 DevConfig

DevConfig (Device Configuration Utility) is the preferred tool for configuring the CR200(X). It is made available as part of LoggerNet, PC400, and at [www.campbellsci.com](http://www.campbellsci.com).

Features of DevConfig include:

- Communicates with devices via direct RS-232 only.
- Sends operating systems to supported device types.
- Sets datalogger clocks and sends program files to dataloggers.
- Identifies operating system types and versions.
- Provides a reporting facility wherein a summary of the current configuration of a device can be shown, printed or saved to a file. The file can be used to restore settings, or set settings in like devices.
- Provides a terminal emulator useful in configuring devices not directly supported by DevConfig's graphical user interface.
- Shows Help as prompts and explanations. Help for the appropriate settings for a particular device can also be found in the user's manual for that device.
- Updates from Campbell Scientific's web site.

As shown in *FIGURE. DevConfig CR200(X) Utility* (p. 60), the DevConfig window is divided into two main sections: the device selection panel on the left side and tabs on the right side. After choosing a device on the left, choose from the list of the serial ports (COM1, COM2, etc.) installed on the PC. A selection of baud rates is offered only if the device supports more than one baud rate. The page for each device presents instructions to set up the device to communicate with DevConfig. Different device types offer one or more tabs on the right.

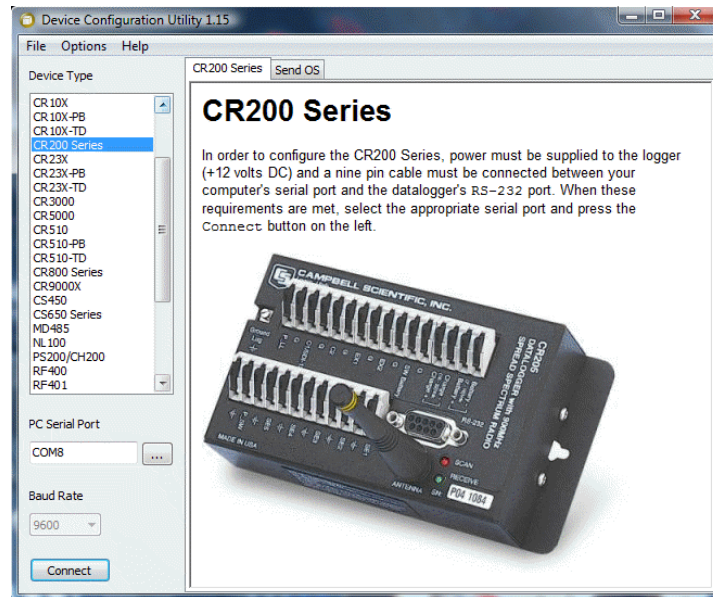


Figure 37: DevConfig Utility

## 8.2 Sending the Operating System

The CR200(X) is shipped with the operating system pre-loaded. However, OS updates are made available at [www.campbellsci.com](http://www.campbellsci.com) and can be sent to the CR200(X).

Since sending an OS to the CR200(X) resets memory, data loss will certainly occur. Depending on several factors, the CR200(X) may also become incapacitated for a time. Consider the following before updating the OS.

1. Is sending the OS necessary to correct a critical problem? -- If not, consider waiting until a scheduled maintenance visit to the site.
2. Is the site conveniently accessible such that a site visit can be undertaken to correct a problem of reset settings without excessive expense?

If the OS must be sent, and the site is difficult or expensive to access, try the OS download procedure on an identically programmed, more conveniently located datalogger.

### 8.2.1 Sending OS with DevConfig

*FIGURE. DevConfig OS Download Window* (p. 61) and *FIGURE. Dialog Box Confirming OS Download* (p. 61) show DevConfig windows displayed during the OS download process.

---

**Caution** Sending an operating system with DevConfig will erase all existing data and reset all settings to factory defaults.

---

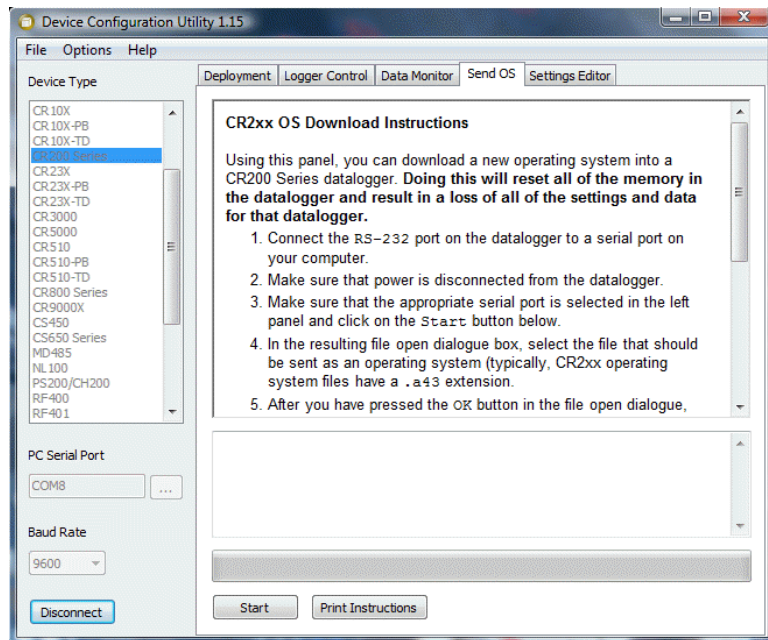


Figure 38: DevConfig OS Download Window

Text in the Send OS tab lists instructions for sending an operating system to the CR200(X).

When the **Start** button is clicked, DevConfig offers a file open dialog box that prompts for the operating system file (\*.obj file). When the CR200(X) is powered-up, DevConfig starts to send the operating system.

When the operating system has been sent, a confirming message dialog box.

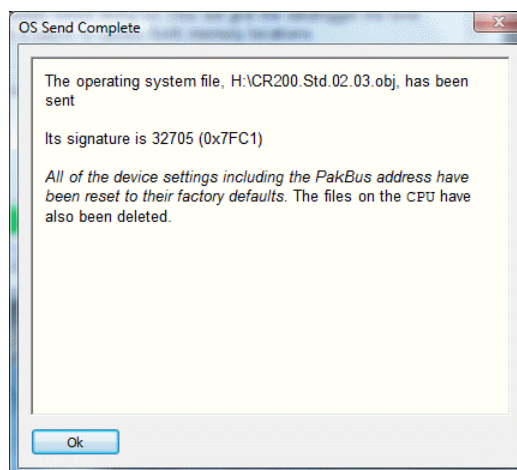


Figure 39: Dialog Box Confirming OS Download

The information in the dialog helps to corroborate the signature of the operating system sent.

## 8.3 Settings

### 8.3.1 Settings via DevConfig

The CR200(X) has a number of properties, referred to as "settings", some of which are specific to the PakBus® communications protocol.

---

**Read More!** PakBus® is discussed in [PakBus® Overview](#) (p. 133) and the PakBus® Networking Guide available at [www.campbellsci.com](http://www.campbellsci.com).

---

DevConfig | Settings Editor tab provides access to most PakBus® settings, however, the DevConfig | Deployment tab makes configuring most of these settings easier.

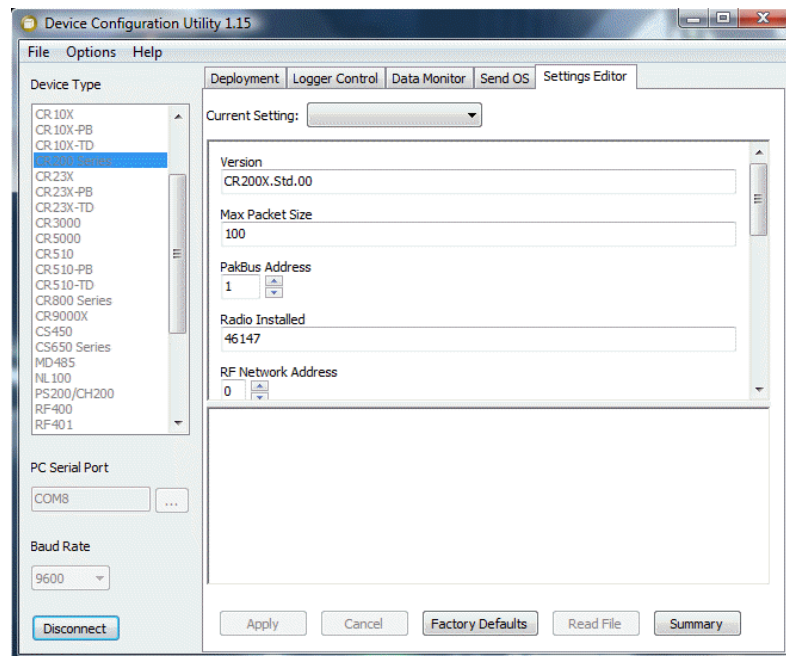


Figure 40: DevConfig Settings Editor

As shown in [FIGURE. DevConfig Settings Editor](#) (p. 62), the top of the Settings Editor is a grid that allows the user to view and edit the settings for the device. The grid is divided into two columns with the setting name appearing in the left hand column and the setting value appearing in the right hand column. Change the currently selected cell with the mouse or by using up-arrow and down-arrow keys as well as the Page-Up and Page-Down keys. When clicking in the setting names column, the value cell associated with that name will automatically be made active. Edit a setting by selecting the value, pressing the F2 key or by

double clicking on a value cell with the mouse. The grid will not allow read-only settings to be edited.

The bottom of the Settings Editor displays help for the setting that has focus on the top of the screen.

Once a setting is changed, click **Apply** or **Cancel**. These buttons will only become enabled after a setting has been changed. If the device accepts the settings, a configuration summary dialogue is shown (*FIGURE. Summary of CR200(X) Configuration* p. 63) that gives the user a chance to save and print the settings for the device.

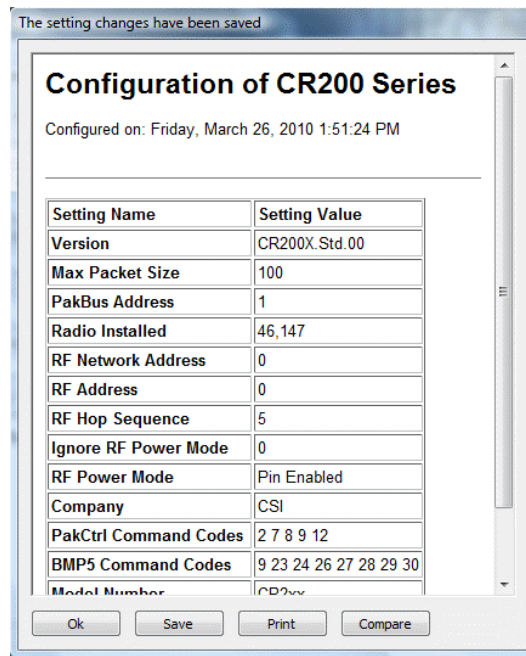


Figure 41: Summary of CR200(X) Configuration

Clicking the **Factory Defaults** button on the Settings Editor will send a command to the device to revert to its factory default settings. The reverted values will not take effect until the final changes have been applied. This button will remain disabled if the device does not support the DevConfig protocol messages.

Clicking **Save** on the summary screen will save the configuration to an XML file. This file can be used to load a saved configuration back into a device by clicking **Read File** and **Apply**.

### 8.3.1.1 Deployment Tab

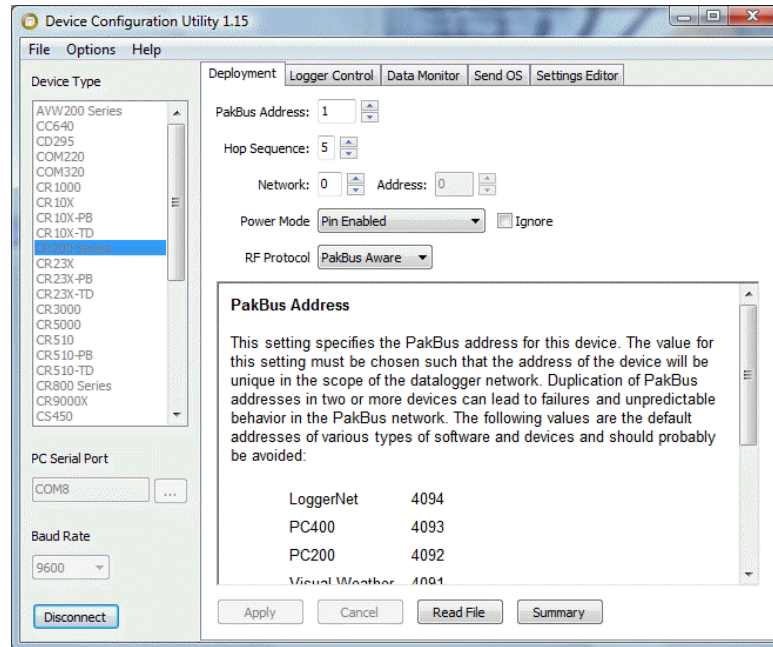


Figure 42: DevConfig Deployment Tab

As shown in [FIGURE. DevConfig Deployment Tab](#) p. 64, the **Deployment** tab allows the user to configure the datalogger prior to deploying it. **Deployment** tab settings can also be accessed through the **Setting Editor** tab and the Status table.

The CR200(X) default PakBus address is 1. Unless the CR200(X) is used in a network, there may be no need to change the PakBus address or any other default setting.

To change the PakBus address, use the up and down arrows next to the PakBus Address field or key in the desired number (e.g., 5) and click the Apply button.

The Spread Spectrum radios in the CR200(X) series, and in the RF401, have address, frequency, and power settings. These addresses are not PakBus addresses but an address the radio encodes in its message. For radios in a PakBus network to talk to each other the address and frequency settings must be the same in ALL radios.

To change radio settings, use the up and down arrows or drop-down menus next to the field of interest and click the Apply button when finished.

- **Hop Sequence** specifies the hopping sequence that will be used for the built-in radio. Spread Spectrum radios have a band of frequencies that they use. The radios “hop” from one frequency to another within this band, allowing multiple sets of radios to communicate at the same time without interfering with each other. This value should be set to match the Hop



Sequence value of the RF400 series base station used to communicate with the CR200(X) so they can contact each other.

- **Network** specifies the radio network address of the built-in radio which is combined with the radio address and sent as part of a packet header with each message. This value should be set to match the network address of the RF400 series base station used to communicate with the CR200(X).
- **Address** specifies the address of the built-in radio which is combined with the network value and sent as part of a packet header with each message. This value should be set to match the address of the RF400 series base station used to communicate with the CR200(X).
- **Power Mode** specifies the power wait state that will be set in the built-in radio and how much power the CR200(X) radio consumes from its power supply. This value should be consistent with the RF400 series base station used to communicate with the CR200(X).
- **RF Protocol** identifies the radio protocol that will be used for the CR200(X). For networks that include older CR205 and RF400 operating systems this will be Transparent. If no devices with older operating systems are present in the network PakBus Aware should be used.
- **Help** is displayed at the bottom of the Deployment tab. When finished, Apply the settings to the datalogger. The Summary window will appear. Save or Print the settings to archive or to use as a template for another datalogger.
- **Cancel** causes the datalogger to ignore the changes. Read File provides the opportunity to load settings saved previously from this or another similar datalogger. Changes loaded from a file will not be written to the datalogger until Apply is clicked.

### 8.3.1.2 Logger Control Tab

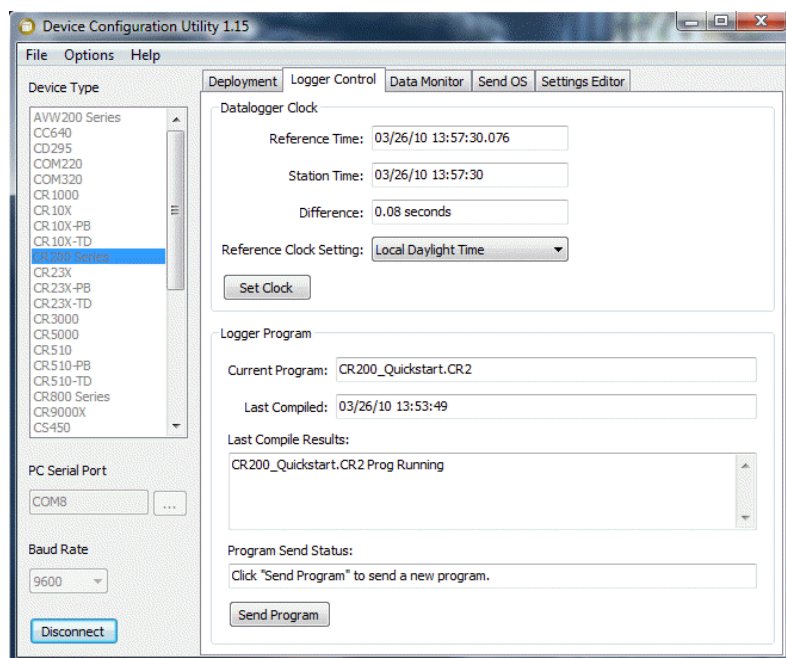


Figure 43: DevConfig Logger Control Tab

- Clocks in the PC and CR200(X) are checked every second and the difference displayed. The **System Clock Setting** allows entering what offset, if any, to use with respect to standard time (Local Daylight Time or UTC, Greenwich mean time). The value selected for this control is remembered between sessions. Clicking the **Set Clock** button will synchronize the station clock to the current computer system time.
- **Current Program** displays the current program known to be running in the datalogger. This value is empty if there is no current program.
- The **Last Compiled** field displays the time when the currently running program was last compiled by the datalogger. As with the Current Program field, this value is read from the datalogger if it is available.
- **Last Compile Results** shows the compile results string as reported by the datalogger.
- The **Send Program** button presents an open file dialog from which to select a program file to be sent to the datalogger. The field above the button is updated as the send operation progresses. When the program has been sent the Current Program, Last Compiled, and Last Compile Results fields are filled in.

### 8.3.2 Settings via CRBASIC

Some variables in the status table can be requested or set during program execution using the CRBASIC SetStatus () command. Entries can be requested or set by setting a Public or Dim variable equivalent to the status table entry, as can be done with variables in any data table. For example, to set a variable, x, equal to a status table entry, the syntax is,

```
x = Status.StatusTableEntry
```

Careful programming is required when changing settings via CRBASIC to ensure users are not inadvertently blocked from communicating with the CR200(X), the remedy for which may be a site visit.

### 8.3.3 Settings via Terminal Emulator

The CR200(X) has a simple terminal interface that can be accessed with a PC running a terminal emulator such as those available in PC200, PC400, or LoggerNet.

Terminal mode is entered while the PC is in telecommunications with the datalogger through a terminal emulator program. It is easily accessed through Campbell Scientific datalogger support software, but is accessible with terminal emulator programs such as Windows Hyperterminal. Datalogger keyboards and displays cannot be used.

The terminal emulator is accessed by navigating to the Datalogger menu in PC200W, the Datalogger menu in the Connect screen of LoggerNet, or the Tools menu in PC400. A serial cable must be connected between the computer/terminal and the CR200(X) in order to establish communications.



The CR200(X) will time out and exit the terminal mode if it does not receive a command within 12 seconds.

Enter a command by pressing the command followed by the carriage return. The commands are:

<b>Commands</b>	<b>Action (what returns)</b>
A	Trap Code 16 Information "OK" = memory ok "2005-7-14 13:56:6=16" = memory failure, 16 denotes trap code 16, date/time is when it occurred.  NOTE: When a trap code 16 happens, the datalogger suspends running the program and the red LED flashes twice every scan interval. A trap code of 16 means that the datalogger's memory needs to be replaced.
1	Get clock
2	Set clock (after setting clock, exits terminal mode)
3	Status codes
4	Status table
5	Public table (real time values)
6	Most recent record from user defined table 1
7	Most recent record from user defined table 2
8	Most recent record from user defined table 3
9	Most recent record from user defined table 4
SDI12	Goes into the SDI-12 terminal mode and remains there

### 8.3.4 Durable Settings

Many CR200(X) settings can be changed remotely over a telecommunications link either directly or as part of the CRBASIC program. This convenience comes with the risk of inadvertently changing settings and disabling communications. Such an instance will likely require an on-site visit to correct the problem. For example, digital cell modems are often controlled by a switched 12 Volt (SW Battery) channel. SW Battery is normally off, so, if the program controlling SW Battery is disabled, such as by replacing it with a program that neglects SW Battery control, the cell modem is switched off and the remote CR200(X) drops out of telecommunications.



# Section 9. Programming

---

## 9.1 Inserting Comments into Program

Comments are non-functioning text placed within the body of a program to document or clarify program algorithms.

As shown in *CRBASIC EXAMPLE. Inserting Comments* (p. 69), comments are inserted into a program by preceding the comment with a single quote ('). Comments can be entered either as independent lines or following CR200(X) code. When the CR200(X) compiler sees a single quote ('), it ignores the rest of the line.

CRBASIC EXAMPLE 1.	Inserting Comments
<i>'Declaration of variables starts here.</i>	
Public Start(6)	<i>'Declare the start time array</i>

## 9.2 Sending Programs

The CR200(X) requires a program be sent to its memory to direct measurement, processing, and data storage operations. Programs are sent with LoggerNet / PC400 / RTDAQ / PC200W datalogger support software.

Programs can also be sent from Device Configuration Utility (DevConfig). A good practice is to always retrieve data from the CR200(X) before sending a program, otherwise, data may be lost.

The CR200(X) does not have an on-board compiler to create the binary (.BIN) program file required by the datalogger. Instead, the datalogger support software creates a binary file using the appropriate compiler. The compiler is chosen to match the version of operating system found in the CR200(X) and a BIN file with the same filename as the selected CR2 file is created and sent to the datalogger.

It is important that the compiler used match the operating system version in the datalogger. If a different compiler is used, the program send will fail. Compilers are typically stored at C:\Campbellsci\Lib\CR200(X)\Compilers.

The most current CR200(X) operating system along with compiler and CRBasic Editor support files may be obtained at [www.campbellsci.com/downloads](http://www.campbellsci.com/downloads).

## 9.3 Writing Programs

Programs are created with either Short Cut or CRBASIC Editor. Short Cut is available at no charge at [www.campbellsci.com](http://www.campbellsci.com). CRBASIC Editor is a program

in the LoggerNet / PC400 datalogger support software suites. Programs can be up to 19.6 KBytes in size although typical programs are smaller.

### 9.3.1 Short Cut Editor and Program Generator

Short Cut is easy-to-use menu-driven software that presents the user with lists of predefined measurement, processing, and control algorithms from which to choose. The user makes choices and Short Cut writes the CRBASIC code required to perform the tasks. Short Cut creates a wiring diagram to simplify connection of sensors and external devices. [Quickstart Tutorial](#) (p. 3) works through a measurement example using Short Cut.

For many complex applications, Short Cut is still a good place to start. When as much information as possible is entered, Short Cut will create a program template from which to work, already formatted with most of the proper structure, measurement routines, and variables. The program can then be edited further using CRBASIC Program Editor.

### 9.3.2 CRBASIC Editor

CR200(X) application programs are written in a variation of BASIC (Beginner's All-purpose Symbolic Instruction Code) computer language, CRBASIC (Campbell Recorder BASIC). CRBASIC Editor is a text editor that facilitates creation and modification of the ASCII text file that constitutes the CR200(X) application program. CRBASIC Editor is available as part of LoggerNet / PC400 / RTDAQ datalogger support software packages.

Fundamental elements of CRBASIC include:

- Variables - named packets of CR200(X) memory into which are stored values that normally vary during program execution. Values are typically the result of measurements and processing. Variables are given an alphanumeric name and can be dimensioned into arrays of related data.
- Constants - discrete packets of CR200(X) memory into which are stored specific values that do not vary during program executions. Constants are given alphanumeric names and assigned values at the beginning declarations of a CRBASIC program.

---

**Note** Keywords and predefined constants are reserved for internal CR200(X) use. If a user programmed variable happens to be a keyword or predefined constant, a runtime or compile error will occur. To correct the error, simply change the variable name by adding or deleting one or more letters, numbers, or the underscore ( \_ ) from the variable name, then recompile and resend the program. CRBASIC Help provides a list of keywords and pre-defined constants.

---

- Common instructions - Instructions and operators used in most BASIC languages, including program control statements, and logic and mathematical operators.
- Special instructions - Instructions unique to CRBASIC, including measurement instructions that access measurement channels, and

processing instructions that compress many common calculations used in CR200(X) dataloggers.

These four elements must be properly placed within the program structure.

## 9.4 Numerical Formats

Four numerical formats are supported by CRBASIC. Most common is the use of base 10 numbers. Scientific notation, binary, and hexadecimal formats may also be used, as shown in [TABLE. Formats for Entering Numbers in CRBASIC](#) (p. 71). Only standard base 10 notation is supported by Campbell Scientific hardware and software displays.

Table 5. Formats for Entering Numbers in CRBASIC		
Format	Example	Base 10 Equivalent Value
Standard	6.832	6.832
Scientific notation	5.67E-8	5.67X10 <sup>-8</sup>
Binary	&B1101	11
Hexadecimal	&HFF	255

Binary format is useful when loading the status (1 = high, 0 = low) of multiple flags or ports into a single variable, e.g., storing the binary number &B11100000 preserves the status of flags 8 through 1. In this case, flags 1 - 5 are low, 6 - 8 are high. [CRBASIC EXAMPLE. Load Binary Information into a Variable](#) (p. 71) shows an algorithm that loads binary status of flags into a LONG integer variable.

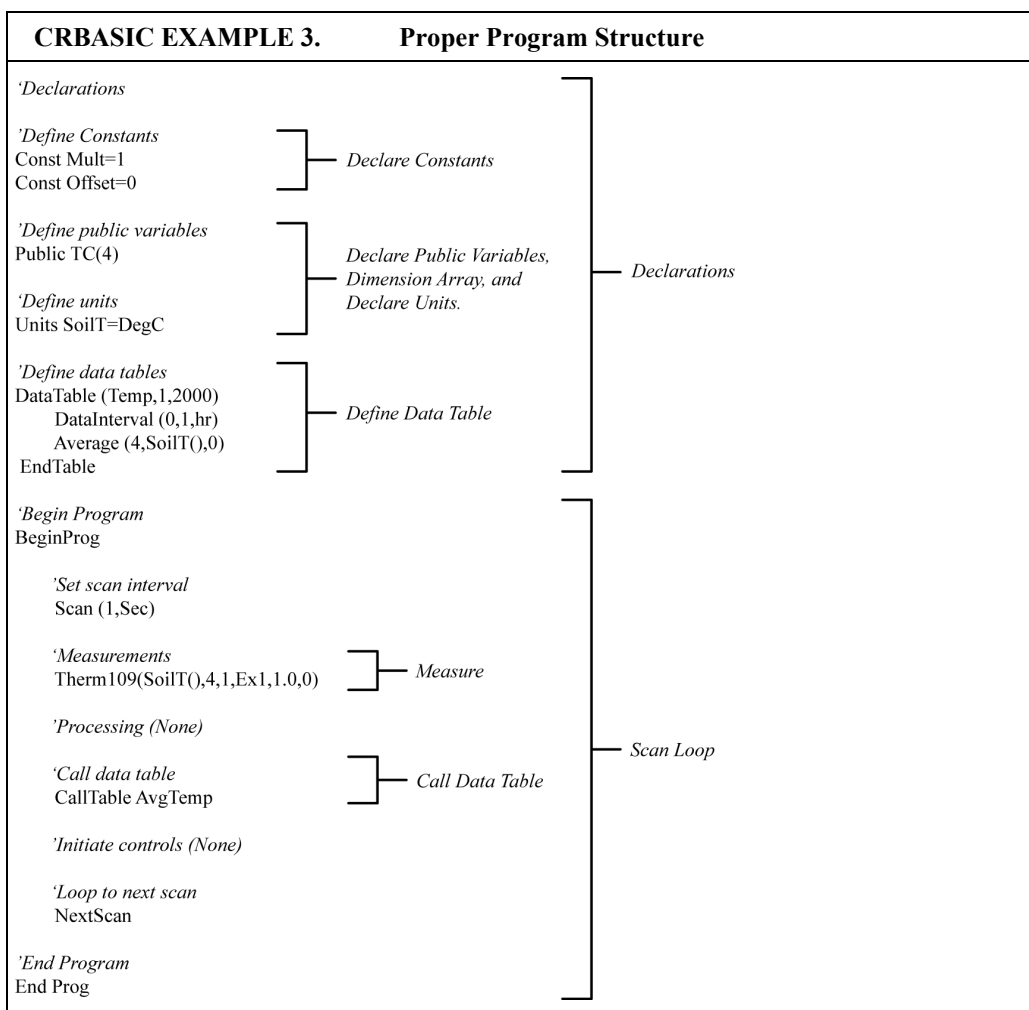
CRBASIC EXAMPLE 2.	Load binary information into a single variable
<pre> Public FlagInt Public Flag(8) Public I  DataTable (FlagOut,True,1000)   Sample (1,FlagInt) EndTable BeginProg   Scan (1,Sec)     FlagInt = 0     For I = 1 To 8       Flag (I) = IIF (Flag(I)= 0,0,-1)       If Flag(I) = true then         FlagInt = FlagInt + 2 ^ (I - 1)       EndIf     Next I     CallTable FlagOut   NextScan EndProg </pre>	

## 9.5 Structure

*TABLE. CRBASIC Program Structure* (p. 72) delineates CRBASIC program structure:

Table 6. CRBASIC Program Structure	
Declarations	Define datalogger memory usage. Declare constants, variables, aliases, units, and data tables.
Declare constants	List fixed constants
Declare Public variables	List / dimension variables viewable during program execution
Dimension variables	List / dimension variables not viewable during program execution.
Define Aliases	Assign aliases to variables.
Define Units	Assign engineering units to variable (optional). Units are strictly for documentation. The CR200(X) makes no use of Units nor checks Unit accuracy.
Define data tables.	Define stored data tables
Process/store trigger	Set triggers when data should be stored. Triggers may be a fixed interval, a condition, or both.
Table size	Set the size of a data table
Processing of Data	List data to be stored in the data table, e.g. samples, averages, maxima, minima, etc.  Processes or calculations repeated during program execution can be packaged in a subroutine and called when needed rather than repeating the code each time.
Begin Program	Begin Program defines the beginning of statements defining datalogger actions.
Set scan interval	The scan sets the interval for a series of measurements
Measurements	Enter measurements to make
Processing	Enter any additional processing
Call Data Table(s)	Declared data tables must be called to process and store data
Initiate controls	Check measurements and initiate controls if necessary
NextScan	Loop back to Set Scan and wait for the next scan
End Program	End Program defines the ending of statements defining datalogger actions.

*CRBASIC EXAMPLE. Proper Program Structure* (p. 73) demonstrates the proper structure of a CRBASIC program.



## 9.6 Declarations I - Single-line Declarations

Public variables, Dim variables, Constants, Units, Aliases, Data Tables and Subroutines are declared at the beginning of a CRBASIC program. [TABLE. Rules for Names](#) (p. 85) lists declaration names and allowed lengths

### 9.6.1 Variables

A variable is a packet of memory, given an alphanumeric name, through which pass measurements and processing results during program execution. Variables are declared either as Public or Dim at the discretion of the programmer. Public variables can be viewed through software numeric monitors. Dim variables cannot. Up to 128 public variables can be declared in a CR200(X) program and up to 48 public variables declared in a CR200 program.

Variable names can be up to 16 characters in length, but most variables should be no more than 12 characters long. This allows for the 4 additional characters that are added as a suffix to the variable name when it is output to a data table.

Variable names cannot start with a number or contain spaces or quote marks ("), but can contain numbers and underscores (\_). Several variables can be declared on a single line, separated by commas:

```
Public RefTemp, AirTemp2, Batt_Volt
```

### 9.6.1.1 Arrays

When a variable is declared, several variables of the same root name can also be declared. This is done by placing a suffix of "(x)" on the alphanumeric name, which creates an array of x number of variables that differ only by the incrementing number in the suffix. For example, rather than declaring four similar variables as follows,

```
Public TempC1
```

```
Public TempC2
```

```
Public TempC3
```

```
Public TempC4
```

simply declare a variable array as shown below:

```
Public TempC(4),
```

This creates in memory the four variables TempC(1), TempC(2), TempC(3), and TempC(4).

A variable array is useful in program operations that affect many variables in the same way. [\*CRBASIC EXAMPLE. Using a Variable Array in Calculations\*](#) (p. 74) shows program code using a variable array to reduce the amount of code required to convert four temperatures from Celsius degrees to Fahrenheit degrees.

In this example a For/Next structure with a changing variable is used to specify which elements of the array will have the logical operation applied to them. The CRBASIC For/Next function will only operate on array elements that are clearly specified and ignore the rest. If an array element is not specifically referenced, e.g., TempC(), CRBASIC references only the first element of the array, TempC(1).



**CRBASIC EXAMPLE 4. Using a variable array in calculations**

```

Public TempC(4)
Public TempF(4)
Dim T

BeginProg
  Scan (1,Sec,0,0)
    Therm109 (TempC(),4,1,Ex1,1.0,0)
    For T = 1 To 4
      TempF(T) = TempC(T) * 1.8 + 32
    Next
  NextScan
EndProg

```

**9.6.1.2 Dimensions**

The CR200(X) cannot use multi-dimensioned arrays.

**9.6.1.3 Data Types**

Variables, calculations, and stored data use IEEE4 4-byte floating point, a binary format, with least significant bit first. Time is stored as integer seconds since midnight, the start of 1990, which is also a 4-byte number.

<i>CR200(X) IEEE4 Data</i>		
Word Size	Range	Resolution
4 bytes	$\pm 1.8 \times 10^{-38}$ to $\pm 1.7 \times 10^{38}$	24 bits (about 7 digits)

**9.6.1.4 Flags**

Flags are a useful program control tool. While any variable can be used as a flag, variables named "Flag" works best because datalogger support software automatically adds variables call "Flag" to the Ports and Flags window. Because the CR200(X) does not support the Boolean data type, the IIF function may be used to distinguish between zero and non-zero values, effectively creating a Boolean value. The value of -1 (all bits on) is defined as true and the value of zero (all bits off) is defined as false. *CRBASIC EXAMPLE. Flag Declaration and Use* (p. 76) shows an example using a flag to initiate measurements.

CRBASIC EXAMPLE 5.	Flag Declaration and Use
<pre> Public batt_volt Public Flag  BeginProg   Scan (1,Sec)     Flag = IIF (Flag=0,0,-1)     If Flag = true Then       Battery (batt_volt)     EndIf   NextScan EndProg </pre>	

## 9.6.2 Constants

*CRBASIC EXAMPLE. Using the Const Declaration* (p. 76) shows use of the constant declaration. A constant can be declared at the beginning of a program to assign an alphanumeric name to be used in place of a value so the program can refer to the name rather than the value itself. Using a constant in place of a value can make the program easier to read and modify, and more secure against unintended changes.

---

**Note** Using all uppercase for constant names may make them easier to recognize.

---

CRBASIC EXAMPLE 6.	Using the Const Declaration
<pre> Public PTempC, PTempF Const CtoF_Mult = 1.8 Const CtoF_Offset = 32  BeginProg   Scan (1,Sec,0,0)     Therm109 (PTempC,1,1,Ex1,1.0,0)     PTempF = PTempC * CtoF_Mult + CtoF_Offset   NextScan EndProg </pre>	

### 9.6.2.1 Predefined Constants

Several words are reserved for use by CRBASIC. These words cannot be used as variable or table names in a program. Predefined constants include some instruction names, as well as valid alphanumeric names for instruction parameters. In general, instruction names should not be used as variable, constant, or table names in a datalogger program, even if they are not specifically listed as a predefined constant. If a predefined constant, such as "Sub" is used as a variable in a program, an error similar to the following may be, but is not always, displayed at CRBASIC pre-compile.

```

Compile Failed!
line 8: Sub is already is use as a predefined CONST.

```

*TABLE. Predefined Constants and Reserved Words* (p. 77) lists predefined constants.

Table 7. Predefined Constants and Reserved Words			
Case	Day	DO	FOR
FALSE	Hr	If	Msec
min	mv2500	mv5000	PROG
SCAN	Select	SUB	Sec
TABLE	TRUE	Usec	Until
EX1	EX2	While	

### 9.6.3 Alias and Unit Declarations

A variable can be assigned a second name, or alias, by which it can be called throughout the program. Aliasing is particularly useful when using arrays. Arrays are powerful features in complex programs, but place the same appellation on a number of variables. The use of an alias allows the power of the array to be used with the clarity of unique names.

Each variable can be assigned units to clarify the meaning. Units are not used elsewhere in programming, but rather add meaning to resultant data table headers.

CRBASIC EXAMPLE 7.	Alias and Unit Declaration
<pre>Public TempC(2)  Alias TempC(1)=AirTempC Alias TempC(2)=SoilTempC  Units TempC()=Deg_C  BeginProg   Scan (1,Sec)     Therm109 (TempC(),2,1,Ex1,1.0,0)   NextScan EndProg</pre>	

## 9.7 Declarations II - Declared Sequences

### 9.7.1 Data Tables

Data are stored in tables as directed by the CRBASIC program. A data table is created by a series of CRBASIC instructions entered after variable declarations but before the BeginProg instruction. These instructions include:

```

DataTable ( )
    Output Trigger Condition(s)
    Output Processing Instructions
EndTable

```

A data table is essentially a file that resides in CR200(X) memory. The file is written to each time data are directed to that file. The trigger that initiates data storage is tripped either by the CR200(X)'s clock, or by an event, such as a high temperature. Up to 8 data tables can be created by the program for a CR200(X) (4 data tables for a CR200). The data tables may store individual measurements, individual calculated values, or summary data such as averages, maxima, or minima to data tables.

Each data table is associated with overhead information that becomes part of the ASCII file header (first few lines of the file) when data are downloaded to a PC. Overhead information includes:

- table format
- datalogger type and operating system version,
- name of the CRBASIC program running in the datalogger
- name of the data table (limited to 16 characters)
- alphanumeric field names to attach at the head of data columns

This information is referred to as "table definitions."

*TABLE. Typical Data Table* (p. 79) shows a data file as it appears after the associated data table has been downloaded from a CR200(X) programmed with the code in *CRBASIC EXAMPLE. Definition and Use of a Data Table* (p. 79). The data file consists of five or more lines. Each line consists of one or more fields. The first four lines constitute the file header. Subsequent lines contain data.

The first header line is the Environment Line. It consists of eight fields, listed in *TABLE. TOA5 Environment Line* (p. 78).

Table 8. TOA5 Environment Line		
Field	Description	Changed via
1	file type (always TOA5)	No Change
2	station name	DevConfig or Program
3	datalogger model	No Change
4	datalogger serial number	No Change
5	datalogger OS version	Send New OS
6	datalogger program name	Send New Program
7	datalogger program signature	Send / Change Program
8	table name	Change Program

The second header line reports field names. This line consists of a set of comma-delimited strings that identify the name of individual fields as given in the datalogger program. If the field is an element of an array, the name will be followed by a comma separated list of subscripts within parentheses that identifies the array index. For example, a variable named values that is declared as an array of four elements in the datalogger program will be represented by four field names: values(1), values(2), values(3), and values(4). Scalar variables will not have array subscripts. There will be one value on this line for each scalar value that is defined by the table. Default fieldnames are a combination of the variable names (or alias) from which data are derived and a three letter suffix. The suffix is an abbreviation of the data process that output the data to storage. For example, “Avg” is the abbreviation for average. If the default fieldnames are not acceptable to the programmer, FieldNames () instruction can be used to customized fieldnames.

The third header line identifies engineering units for that field of data. These units are declared in the “Define Units” section of the CRBASIC program, as shown in [CRBASIC EXAMPLE. Definition and Use of a Data Table](#) (p. 79). Units are strictly for documentation.

Subsequent lines are observed data and associated record keeping. The first field being a timestamp, the second the record (data line) number.

---

**Read More!** See [TABLE. Abbreviations of Names of Data Processes](#) (p. 90) for a list of default field names.

---

<b>Table 9. Typical Data Table</b>							
TOA5	CR200(X)	CR2xx	No_SN	CR200(X).Std.01	Data.CR2	4098	OneMin
TIMESTAMP	RECORD	BattV_Min	T109_C_Avg	Rain_mm_Tot			
TS	RN	Volts	Deg C	Mm			
		Min	Avg	Tot			
12/30/2010 12:59:00 PM	0	12.52359	-7.390147	0			
12/30/2010 1:00:00 PM	1	12.56724	-6.412211	0			
12/30/2010 1:01:00 PM	2	12.52695	-6.267832	0			
12/30/2010 1:02:00 PM	3	12.56472	-6.298645	0			

CRBASIC EXAMPLE 8.	Definition and Use of a Data Table
	<pre> 'Declare Variables Public Batt_Volt Public T109_C(2)  'Define Units Units Batt_Volt=Volts Units T109_C(2)=Deg C  'Define Data Tables DataTable (OneMin,True,-1)     DataInterval (0,1,Min)     Average (1,Batt_Volt,False)     Average (2,T109_C(1),False) EndTable  DataTable (Table1,True,-1)     DataInterval (0,1440,Min)     Minimum (1,Batt_Volt,False,False) EndTable  'Main Program BeginProg     Scan (5,Sec)          'Default Datalogger Battery Voltage measurement Batt_Volt:         Battery (Batt_Volt)          '109-L Thermistor measurements Temp_C:         Therm109(T109_C(),2,1,Ex1,1,0)          'Call Data Tables and Store Data         CallTable (OneMin)         CallTable (Table1)      NextScan EndProg </pre>

As shown in [CRBASIC EXAMPLE. Definition and Use of a Data Table](#) (p. 79), data table declaration begins with the `DataTable ()` instruction and ends with the `EndTable ()` instruction. Between `DataTable ()` and `EndTable ()` are instructions that define what data to store and under what conditions data are stored. A data table must be called by the CRBASIC program for data storage processing to occur. Typically, data tables are called by the `CallTable ()` instruction once each Scan.

### 9.7.1.1 DataTable () and EndTable () Instructions

The `DataTable` instruction has three parameters: a user-specified alphanumeric name for the table (e.g., "OneMin"), a trigger condition (e.g., "True"), and the size to make the table in RAM (e.g., auto allocated).

- **Name**-The table name can be any combination of numbers and letters up to 16 characters in length. The first character must be a letter.

- **TrigVar**-Controls whether or not data records are written to storage. Data records are written to storage if TrigVar is true and if other conditions, such as DataInterval (), are met. Default setting is -1 (True). TrigVar may be a variable, expression, or constant. TrigVar does not control intermediate processing. Intermediate processing is controlled by the disable variable, DisableVar, which is a parameter in all output processing instructions (see [Output Processing Instructions](#) (p. 81)).

---

**Read More!** [TrigVar and DisableVar - Controlling Data Output and Output Processing](#) (p. 125) discusses the use of TrigVar and DisableVar in special applications.

---

- **Size**-Table size is the number of records to store in a table before new data begins overwriting old data. If "10" is entered, 10 records are stored in the table -- the eleventh record will overwrite the first record. If "-1" is entered, memory for the table is automatically allocated at the time the program compiles. Auto allocation is preferred in most applications since the CR200(X) sizes all tables such that they fill (and begin overwriting the oldest data) at about the same time. Approximately 2K bytes of extra data table space is allocated to minimize the possibility of new data over writing the oldest data in ring memory when support software collects the oldest data at the same time new data are written. These extra records are not reported in the Status Table and are not reported to the support software and so are not collected.

*CRBASIC EXAMPLE. Definition and Use of a Data Table* (p. 79) creates a data table named "OneMin", stores data once a minute as defined by DataInterval (), and retains the most recent records in SRAM, up to the automatically allocated memory limit. DataRecordSize entries in the status table report allocated memory in terms of number of records the tables hold.

### 9.7.1.2 DataInterval () Instruction

DataInterval () instructs the CR200(X) to write data records at the specified interval. The interval is independent of the Scan () / NextScan interval; however, it must be a multiple of the Scan () / NextScan interval. The data interval must be at least one minute.

DataInterval does not override the trigger condition in the DataTable instruction. If the trigger is not set always true by entering a constant, it is a condition that must be met in addition to the time interval before data will be stored.

### 9.7.1.3 Output Processing Instructions

Data storage processing ("output processing") instructions determine what data are stored in the data table. When a data table is called in the CRBASIC program, data storage processing instructions process variables holding current inputs or calculations. If trigger conditions are true, e.g. the required interval has expired, processed values are stored ("output") in the data table. In *CRBASIC EXAMPLE. Definition and Use of a Data Table* (p. 79), three averages are stored.

Consider the Average () instruction as an example of output processing instructions. Average () stores the average of a variable over the data storage output interval. Its parameters are:

- **Reps**-number of elements in the variable array for which to calculate averages. Reps is set to 1 to average Batt\_Volt, and set to 2 to average 2 thermistor temperatures, both of which reside in the variable array "T109\_C".
- **Source**-variable array to average. Variable arrays Batt\_Volt (an array of 1) and T109\_C() (an array of 2) are used.
- **DisableVar**-controls whether or not a measurement or value is included in an output processing function. A measurement or value will not be included if the disable variable is true ( $\neq 0$ ). For example, in the case of an Average () output processing instruction, if, on a particular pass through the data table, the Average () disable variable true, the value resident in the variable to be averaged with not be included in the average. Program CRBASIC EXAMPLE. Use of the Disable Variable has "False" entered for the disable variable, so all readings are included in the averages; the average of variable "Oscillator" does not include samples occurring when Flag 1 is high, producing an average of 2, whereas, when Flag 1 is low (all samples used), an average of 1.5 is calculated.

---

Read More! [TrigVar and DisableVar - Controlling Data Output and Output Processing](#) (p. 125) discusses the use of TrigVar and DisableVar in special applications.

---

#### CRBASIC EXAMPLE 9. Use of the Disable Variable

```
'Declare Variables and Units
Public Oscillator
Public Flag
Public DisableVar

'Define Data Tables
DataTable (OscAvgData,True,-1)
  DataInterval (0,1,Min)
  Average (1,Oscillator,DisableVar)
EndTable

'Main Program
BeginProg
  Scan (1,Sec)

  'Reset and Increment Counter
  If Oscillator = 2 Then Oscillator = 0
  Oscillator = Oscillator + 1

  'Process and Control
  If Oscillator = 1
```



```

        If Flag = True Then
            DisableVar = True
        End If
    Else
        DisableVar = False
    EndIf

    'Call Data Tables and Store Data
    CallTable (OscAvgData)

NextScan
EndProg

```

---

**Read More!** For a complete list of output processing instructions, see [Data Storage Output Processing](#) (p. 94).

---

## 9.7.2 Subroutines

Subroutines allow a section of code to be called by multiple processes in the main body of a program. Subroutines are defined before the main program body of a program. Program [CRBASIC EXAMPLE. Use of a Subroutine](#) p. 83 shows the use of a subroutine to repeatedly perform a calculation.

### CRBASIC EXAMPLE 10. Use of a Subroutine

```

'Declare Variables and Units
Public Temp(4), I, Temp_F(4)

'Subroutine to convert temperature in degrees C to degrees F
Sub ConvertCtoF
    For I = 1 to 4
        Temp_F = Temp(I)*1.8 + 32
    Next I
EndSub

'Main Program
BeginProg
    Scan (1,Sec)
    Therm109 (Temp(),4,1,Ex1,1.0,0)
    'convert Temperatures to F using Subroutine:
    Call ConvertCtoF
    NextScan
EndProg

```

## 9.8 Program Execution Timing

CR200(X) programs are built within a Scan () / NextScan structure, with only variable and data table declarations outside the Scan () / NextScan structure. In these programs, Scan () / NextScan creates an infinite loop, each periodic pass through the loop being synchronized to the CR200(X) clock. Scan () parameters allow modification of the period. As shown in [CRBASIC EXAMPLE. BeginProg / Scan / NextScan / EndProg Syntax](#) (p. 84), aside from declarations, the CRBASIC program may be relatively short.

Scan () determines how frequently instructions in the program are executed.  
Scan has two parameters:

- Interval is the interval between scans.
- Units is the time unit for the interval. Interval is 1sec <= Interval <= 1 day.

CRBASIC EXAMPLE 11.	BeginProg / Scan / NextScan / EndProg Syntax
<pre>BeginProg   Scan (1,Sec)     Therm109 (TempC(),2,1,Ex1,1.0,0)     CallTable Temp   NextScan EndProg</pre>	

## 9.9 Instructions

In addition to BASIC syntax, additional instructions are included in CRBASIC to facilitate measurements and store data. *CRBASIC Programming Instructions* (p. 93) contains a comprehensive list of these instructions.

### 9.9.1 Measurement and Data Storage Processing

CRBASIC instructions have been created for making measurements and storing data. Measurement instructions set up CR200(X) hardware to make measurements and store results in variables. Data storage instructions process measurements into averages, maxima, minima, standard deviation, FFT, etc.

Each instruction is a keyword followed by a series of informational parameters needed to complete the procedure. For example, the instruction used to set an excitation channel to a specified value is:

```
ExciteV (ExChan,ExmV)
```

"ExciteV" is the keyword. Two parameters follow: *ExChan*, the excitation channel number to which the voltage should be applied; and *ExmV*, is the excitation, in millivolts, to apply to the sensor. The syntax for applying 5000 millivolts to a sensor through excitation channel number 1 is shown in *CRBASIC EXAMPLE. Measurement Instruction Syntax* (p. 84).

CRBASIC EXAMPLE 12.	Measurement Instruction Syntax
<pre>ExciteV (1,mV5000)</pre>	

## 9.9.2 Parameter Types

Many instructions have parameters that allow different types of inputs. Common input type prompts are listed below. Allowed input types are specifically identified in the description of each instruction in CRBASIC Editor Help.

- Constant, or Expression that evaluates as a constant
- Variable
- Variable or Array
- Constant, Variable, or Expression
- Constant, Variable, Array, or Expression
- Name
- Name or list of Names
- Variable, or Expression
- Variable, Array, or Expression

## 9.9.3 Names in Parameters

*TABLE. Rules for Names* (p. 85) lists the maximum length and allowed characters for the names for Variables, Arrays, Constants, etc. The CRBASIC Editor pre-compiler will identify names that are too long or improperly formatted.

<b>Table 10. Rules for Names</b>		
<b>Name for</b>	<b>Maximum Length (number of characters)</b>	<b>Allowed characters</b>
Variable or Array	16	Letters A-Z, upper or lower case, underscore "_", and numbers 0-9. The name must start with a letter. CRBASIC is not case sensitive
Constant	16	
Units	10	
Alias	16	
Station Name	16	
Data Table Name	16	
Field name	16	
Field Name Description	58	

## 9.9.4 Expressions in Parameters

**Read More!** See [Expressions](#) (p. 87) for more information on expressions.

Many parameters allow the entry of expressions. If an expression is a comparison, it will return -1 if the comparison is true and 0 if it is false ([Logical Expressions](#) (p. 88)). *CRBASIC EXAMPLE. Use of Expressions in Parameters* (p. 86) shows an example of the use of expressions in parameters in the DataTable instruction, where the trigger condition is entered as an expression. Suppose the variable TC is a thermistor temperature:

CRBASIC EXAMPLE 13.      Use of Expressions in Parameters
<pre>'DataTable (Name, TrigVar, Size) DataTable (Temp, TC &gt; 100, 5000)</pre>

When the trigger is "TC > 100", a TC temperature > 100 will set the trigger to true and data is stored.

## 9.9.5 Arrays of Multipliers and Offsets

A single measurement instruction can measure a series of sensors and apply individual calibration factors to each sensor as shown in [CRBASIC EXAMPLE. Use of Arrays as Multipliers and Offsets](#) (p. 86). Storing calibration factors in variable arrays, and placing the array variables in the multiplier and offset parameters of the measurement instruction, makes this possible. The measurement instruction uses repetitions to implement this feature by stepping through the multiplier and offset arrays as it steps through the measurement input channels. If the multiplier and offset are not arrays, the same multiplier and offset are used for each repetition.

CRBASIC EXAMPLE 14.      Use of Arrays as Multipliers and Offsets
<pre>Public Pressure(3), Mult(3), Offset(3)  DataTable (AvgPress,1,-1)   DataInterval (0,60,Min)   Average (3,Pressure(),IEEE4,0) EndTable  BeginProg   'Calibration Factors:   Mult(1)=0.123 : Offset(1)=0.23   Mult(2)=0.115 : Offset(2)=0.234   Mult(3)=0.114 : Offset(3)=0.224    Scan (1,Sec)     'VoltSe instruction using array of multipliers and offsets:     VoltSe (Pressure(),3,1,Mult(),Offset())     CallTable AvgPress   NextScan EndProg</pre>

---

**Read More!** More information is available in CRBASIC Editor Help topic "Multipliers and Offsets with Repetitions".

---

## 9.10 Expressions

An expression is a series of words, operators, or numbers that produce a value or result. Expressions are evaluated expression from left to right, with deference to precedence rules.

Two types of expressions, mathematical and programming, are used in CRBASIC. A useful property of expressions in CRBASIC is that they are equivalent to and often interchangeable with their results.

Consider the expressions:

$x = (z * 1.8) + 32$  (a mathematical expression)

If  $x = 23$  then  $y = 5$  (programming expression)

The variable  $x$  can be omitted and the expressions combined and written as:

If  $(z * 1.8 + 32 = 23)$  then  $y = 5$

Replacing the result with the expression should be done judiciously and with the realization that doing so may make program code more difficult to decipher.

### 9.10.1 Floating Point Arithmetic

Variables and calculations are performed internally in single precision IEEE4 4-byte floating point, a binary format.

Floating point arithmetic is common in many electronic computational systems, but it has pitfalls high-level programmers should be aware of. Several sources discuss floating point arithmetic thoroughly. One readily available source is the topic "Floating Point" at Wikipedia.org. In summary, CR200(X) programmers should consider at least the following:

- Floating point numbers do not perfectly mimic real numbers.
- Floating point arithmetic does not perfectly mimic true arithmetic.
- Avoid use of equality in conditional statements. Use  $\geq$  and  $\leq$  instead. For example, use "If  $X \Rightarrow Y$ , then do" rather than using, "If  $X = Y$ , then do".

### 9.10.2 Mathematical Operations

Mathematical operations are written out much as they are algebraically. For example, to convert Celsius temperature to Fahrenheit, the syntax is:

```
TempF = TempC * 1.8 + 32
```

*CRBASIC EXAMPLE. Use of Variable Arrays to conserve Code* p. 88 shows example code to convert five temperatures in a variable array from C to F:

CRBASIC EXAMPLE 15.      Use of Variable Arrays to Conserve Code Space
<pre> For I = 1 to 5   TempC(I) = TempC(I) * 1.8 + 32 Next I </pre>

### 9.10.3 Logical Expressions

Measurements can indicate absence or presence of an event. For example, an RH measurement of 100% indicates a condensation event such as fog, rain, or dew. The CR200(X) can render events into binary form for further processing, i.e., events can either be TRUE, indicating the condition occurred or is occurring, or FALSE, indicating the condition has not yet occurred or is over.

*Several words are commonly interchanged with True / False such as High / Low, On / Off, Yes / No, Set / Reset, Trigger / Do Not Trigger. The CR200(X) understands only True / False or -1 / 0, however. The CR200(X) represents "true" with "-1" because AND / OR operators are the same for logical statements and binary bitwise comparisons.*

*In the binary number system internal to the CR200(X), "-1" is expressed with all bits equal to 1 (11111111). "0" has all bits equal to 0 (00000000). When -1 is ANDed with any other number, the result is the other number. This ensures that if the other number is non-zero (true), the result is non-zero.*

Using TRUE or FALSE conditions with logic operators such as AND and OR, logical expressions can be encoded into a CR200(X) to perform three general logic functions, facilitating conditional processing and control applications.

1. Evaluate an expression, then take one path or action if the expression is true (= -1), and / or another path or action if the expression is false (= 0).
2. Evaluate multiple expressions linked with AND or OR.
3. Evaluate multiple and / or links.

The following commands and logical operators are used to construct logical expressions. *CRBASIC EXAMPLE. Logical Expression Examples* p. 89 demonstrate some logical expressions.

- IF
- AND
- OR
- NOT
- XOR
- IIF

<b>Table 11. Logical Expression Examples</b>	
If X >= 5 then Y = 0	
Sets the variable Y to 0 if the expression "X >= 5" is true, i.e. if X is greater than or equal to 5. The CR200(X) evaluates the expression (X >= 5) and registers in system memory a -1 if the expression is true, or a 0 if the expression is false.	
If X >= 5 OR Z = 2 then Y = 0	
Sets Y = 0 if either X >= 5 or Z = 2 is true.	
If X >= 5 AND Z = 2 then Y = 0	
Sets Y = 0 only if both X >= 5 and Z = 2 are true.	
If 6 then Y = 0.	
"If 6" is true since "6" (a non-zero number) is returned, so Y is set to 0 every time the statement is executed.	
If 0 then Y = 0.	
"If 0" is false since "0" is returned, so Y will never be set to 0 by this statement.	
Z = (X > Y) .	
Z will equal -1 if X > Y, or Z will equal 0 if X <= Y.	
<p>The NOT operator simply complements every bit in the word. A Boolean can only be 0 or have all of its bits set to 1. Complementing a Boolean will turn TRUE (all bits set) to FALSE (all bits complemented to 0). Since the CR200(X) cannot declare a variable to be a Boolean data type, IIF is used to force the logical results to either 0 or -1.</p> <p>Example Program</p> <pre>'(a AND b) = (26 AND 26) = (&amp;b11010 AND &amp;b11010) = '&amp;b11010. NOT (&amp;b11010) yields &amp;b00101.  'This is non-zero, so when converted to a 'BOOLEAN, it becomes TRUE. Public a Public b Public is_true Public not_is_true Public not_a_and_b BeginProg   a = 26   b = a   Scan (1,Sec)     is_true = IIF(a AND b = 0,0,-1)     not_is_true = IIF(NOT (is_true) = 0,0,-1)     not_a_and_b = IIF(NOT (a AND b) = 0,0,-1)   NextScan EndProg</pre> <p>'This evaluates to TRUE. 'This evaluates to FALSE. 'This evaluates to TRUE!</p>	

## 9.11 Program Access to Data Tables

CRBASIC has syntax provisions facilitating access to data in tables or information relating to a table. The syntax is entered directly into the CRBASIC program through a variable name. The general form is:

"TableName.FieldName\_Prc (Fieldname Index, Records Back)".

Where:

- **TableName:** name of the data table
- **FieldName:** name of the variable from which the processed value is derived
- **Prc:** Abbreviation of the name of the data process used. See [TABLE. Abbreviations of Names of Data Processes](#) p. 90 for a complete list of these abbreviations-not needed for values from Status or Public tables.
- **Fieldname Index:** Array element number (optional)
- **Records Back:** How far back into the table to go to get the value (optional)

Table 12. Abbreviations of Names of Data Processes	
<b>Abbreviation</b>	<b>Process Name</b>
Tot	Totalize
Avg	Average
Max	Maximum
Min	Minimum
Std	Standard Deviation
	Sample
WVc	WindVector
ETsz	ET
RSO	Solar Radiation (from ET)
TMx	Time of Max
TMn	Time of Min

For instance, to access the number of watchdog errors, use the "Status.WatchDogCnt," where "Status" is the table name, and "WatchDogCnt" is the field name.



Five special variable names are used to access information about a table:

- `FieldName`
- `Output`
- `Record`
- `TableSize`
- `TimeStamp`

Consult CRBASIC Editor Help Index topic "DataTable access" for complete information.



# Section 10. *CRBASIC Programming Instructions*

---

**Read More!** Parameter listings, application information, and code examples are available in CRBASIC Editor Help. CRBASIC Editor is part of LoggerNet / PC400 / RTDAQ.

Select instructions are explained more fully, some with example code, in [Programming Resource Library](#) (p. 109). Example code is throughout the CR200(X) manual. Refer to the table of contents Example index.

---

## 10.1 Program Declarations

### 10.1.1 Variable Declarations & Modifiers

#### **Alias**

Assigns a second name to a variable.

##### Syntax

```
Alias [variable] = [alias name]
```

#### **Dim**

Declares and dimensions private variables. Dimensions are optional.

##### Syntax

```
Dim [variable name (x,y,z)]
```

#### **Public**

Declares and dimensions public variables. Dimensions are optional.

##### Syntax

```
Public [variable name (x,y,z)]
```

#### **Units**

Assigns a unit name to a field associated with a variable.

##### Syntax

```
Units [variable] = [unit name]
```

### 10.1.2 Constant Declarations

#### **Const**

Declares symbolic constants for use in place of numeric entries.

##### Syntax

```
Const [constant name] = [value or expression]
```

## 10.2 Data Table Declarations

### **DataTable ... EndTable**

Mark the beginning and end of a data table.

#### **Syntax**

```
DataTable (Name, TrigVar, Size)
    [data table modifiers]
    [on-line storage destinations]
    [output processing instructions]
EndTable
```

### 10.2.1 Data Table Modifiers

#### **DataInterval**

Sets the time interval for an output table.

#### **Syntax**

```
DataInterval (TintoInt, Interval, Units,)
```

### 10.2.2 Data Storage Output Processing

#### **FieldNames**

Immediately follows an output processing instruction to change default field names.

#### **Syntax**

```
FieldNames ("Fieldname1 : Description1, Fieldname2
: Description2...")
```

#### 10.2.2.1 Single-Source

##### **Average**

Stores the average value over the output interval for the source variable or each element of the array specified.

#### **Syntax**

```
Average (Reps, Source, DisableVar)
```

##### **Maximum**

Stores the maximum value over the output interval.

#### **Syntax**

```
Maximum (Reps, Source, DisableVar, Time)
```

##### **Minimum**

Stores the minimum value over the output interval.

#### **Syntax**

```
Minimum (Reps, Source, DisableVar, Time)
```

##### **Sample**

Stores the current value at the time of output.

#### **Syntax**

```
Sample (Reps, Source)
```

**StdDev**

Calculates the standard deviation over the output interval.

## Syntax

```
StdDev (Reps, Source, DisableVar)
```

**Totalize**

Sums the total over the output interval.

## Syntax

```
Totalize (Reps, Source, DisableVar)
```

**10.2.2.2 Multiple-Source****ETo**

Stores evapotranspiration (ETo) and other meteorological data. Most suitable for output intervals of less than 24 hours.

## Syntax

```
ETo (Temp, RH, u2, Rs, Longitude, Latitude,  
Altitude, DisableVar)
```

**EToDaily**

Stores evapotranspiration (ETo) and other meteorological data. Most suitable for output intervals 24 hours or more.

## Syntax

```
EToDaily (Temp, RH, u2, Rs, Longitude, Latitude,  
Altitude, DisableVar)
```

**WindVector**

Processes wind speed and direction from either polar or orthogonal sensors. To save processing time, only calculations resulting in the requested data are performed.

## Syntax

```
WindVector (Repetitions, Speed/East,  
Direction/North, DisableVar, SensorType,  
WVOutputOpt)
```

---

**Read More!** See [Wind Vector](#) (p. 120).

---

**10.3 Single Execution at Compile****Sub, ExitSub, EndSub**

Declares the name, variables, and code that form a Subroutine. Argument list is optional. ExitSub is optional.

## Syntax

```
Sub subname (argument list)
```

```
[statement block]
```

```
ExitSub
```

```
[statement block]
```

```
EndSub
```

## 10.4 Program Control Instructions

### 10.4.1 Common Controls

#### **BeginProg ... EndProg**

Mark the beginning and end of a program.

Syntax

```
BeginProg
    Program Code
EndProg
```

#### **Call**

Transfers program control from the main program to a subroutine.

Syntax

```
Call subname
```

#### **CallTable**

Calls a data table, typically for output processing.

Syntax

```
CallTable [TableName]
```

#### **Delay**

Delays the program.

Syntax

```
Delay (Delay, Units)
```

#### **Do ... While ... Until ... ExitDo ... Loop**

Repeats a block of statements while a condition is true or until a condition becomes true.

Syntax

```
Do [{While | Until} condition]
    [statementblock]
[ExitDo]
    [statementblock]
Loop
-or-
Do
    [statementblock]
[ExitDo]
    [statementblock]
Loop [{While | Until} condition]
```

#### **For ... To ... Step ... ExitFor ... Next**

Repeats a group of instructions a specified number of times.

Syntax

```
For counter = start To end [ Step increment ]
    [statementblock]
[ExitFor]
    [statementblock]
Next [counter [, counter][, ...]]
```

**If ... Then ... Else ... ElseIf ... EndIf**

Allows conditional execution, based on the evaluation of an expression. Else is optional. ElseIf is optional (EndSelect and EndIf call the same function).

**Syntax**

```
If [condition] Then [thenstatements] Else [elsestatements]
-or-
```

```
If [condition 1] Then
    [then statements]
ElseIf [condition 2] Then
    [elseif then statements]
Else
    [else statements]
EndIf
```

**InterruptSequence**

Specifies code to run when an interrupt condition occurs.

**Syntax**

```
InterruptSequence
```

**LoggerIdentify**

Sets an identification string in the datalogger that will be returned when another string is sent to the datalogger.

**Syntax**

```
LoggernetIdentify (RequestString",ReturnString")
```

**Scan ... NextScan**

Establishes the program scan rate. ExitScan and ContinueScan are optional.

**Syntax**

```
Scan (Interval, Units)
...
Next Scan
...
ContinueScan
...
Next Scan
```

**ScanLEDOff**

Turns off the LED on the datalogger's front case that indicates a program scan is occurring.

**Select Case ... Case ... Case Is ... Case Else ... EndSelect**

Executes one of several statement blocks depending on the value of an expression. CaseElse is optional. (EndSelect and EndIf call the same CR200(X) function).

**Syntax**

```
Select Case testexpression
    Case [expression 1]
        [statement block 1]
    Case [expression 2]
        [statement block 2]
    Case Is [expression fragment]
    Case Else
        [statement block 3]
EndSelect
```

### **While...Wend**

Execute a series of statements in a loop as long as a given condition is true.

#### **Syntax**

```
While Condition
    [StatementBlock]
Wend
```

## **10.5 Measurement Instructions**

### **10.5.1 Diagnostics**

#### **Battery**

Measures input voltage.

#### **Syntax**

```
Battery (Dest)
```

### **10.5.2 Voltage**

#### **VoltSe**

Measures the voltage at a single-ended input with respect to ground.

#### **Syntax**

```
VoltSe (Dest, Reps, SEChan, Mult, Offset)
```

---

**Read More!** See [Bridge Resistance Measurements](#) (p. 41).

---

#### **ExDelSE**

Applies an excitation voltage, delays for a specified period of time, and makes a single-ended voltage measurement. Allowable excitation voltages are +2500mV and +5000mV.

#### **Syntax**

```
ExDelSE (Dest, Reps, SEChan, ExChan, ExmV, Delay,
        Mult, Offset)
```

#### **Therm109**

Measures a Campbell Scientific 109 thermistor

#### **Syntax**

```
Therm109 (Dest, Reps, SEChan, Vx/ExChan,
        SettlingTime, Integ, Mult, Offset)
```

### **10.5.3 Pulse**

---

**Read More!** See [Pulse Count Measurement](#) (p. 42).

---

---

**Note** Pull-up resistors are required when using digital I/O (control) ports for pulse input ([Pulse Input on Digital I/O Channels C1 - C2](#) (p. 45)).

---



**PulseCount**

Measures number or frequency of voltages pulses on a pulse channel.

## Syntax

```
PulseCount (Dest, PChan, PConfig, POption, Mult,
            Offset)
```

**10.5.4 Digital I/O****AnalogPortGet**

Configures an analog port as a digital input and stores the status of the input in a variable.

## Syntax

```
AnalogPortGet (Dest, Port)
```

**AnalogPortSet**

Configures an analog port as a digital output and sets the port either high or low.

## Syntax

```
AnalogPortSet (Port, State)
```

**PeriodAvg**

Measures the period of a signal on any single-ended voltage input channel.

## Syntax

```
PeriodAvg (Dest, SEChan, PAOption, Cycles,
            Timeout, Port, Mult, Offset)
```

**PortGet**

Reads the status of a control port.

## Syntax

```
PortGet (Dest, Port)
```

**PortSet**

Sets the specified port high or low.

## Syntax

```
PortSet (Port, State)
```

**10.5.5 SDI-12**


---

**Read More!** See [\*SDI-12 Sensor Support\*](#) (p. 112).

---

**SDI12Recorder**

The SDI12Recorder instruction is used to retrieve the results from an SDI-12 sensor.

## Syntax

```
SDI12Recorder (Dest, Outstring, Multiplier,
               Offset)
```

**SDI12SensorSetup**

Sets up the datalogger to act as an SDI12 sensor. Used together with SDI12SensorResponse.

### **SDI12SensorResponse**

Holds the source of the data to send to the SDI12 recorder.

#### **Syntax**

```
SDI12SensorSetup (Repetitions, SDIPort, SDIAddress,  
                  ResponseTime)  
SDI12SensorResponse (SDI12Source)
```

## **10.6 Processing and Math Instructions**

### **10.6.1 Mathematical Operators**

- ^** Raise to Power.
- \*** Multiply
- /** Divide
- +** Add
- Subtract
- =** Equals
- <>** Not Equal
- >** Greater Than
- <** Less Than
- >=** Greater Than or Equal
- <=** Less Than or Equal

### **10.6.2 Logical Operators**

#### **AND**

Used to perform a logical conjunction on two expressions.

#### **Syntax**

```
result = expr1 AND expr2
```

#### **IIF**

Evaluates a variable or expression and returns one of two results based on the outcome of that evaluation.

#### **Syntax**

```
Result = IIF (Expression, TrueValue, FalseValue)
```

#### **NOT**

Performs a logical negation on an expression.

#### **Syntax**

```
result = NOT expression
```

**OR**

Used to perform a logical disjunction on two expressions.

Syntax

```
result = expr1 OR expr2
```

**XOR**

Performs a logical exclusion on two expressions.

Syntax

```
result = expr1 XOR expr2
```

## 10.6.3 Trigonometric Functions

### 10.6.3.1 Derived Functions

*TABLE. Derived Trigonometric Functions* (p. 101) is a list of trigonometric functions that can be derived from functions intrinsic to CRBASIC.

<b>Table 13. Derived Trigonometric Functions</b>	
<b>Function</b>	<b>CRBASIC Equivalent</b>
Secant	$\text{Sec} = 1 / \text{Cos}(X)$
Cosecant	$\text{Cosec} = 1 / \text{Sin}(X)$
Cotangent	$\text{Cotan} = 1 / \text{Tan}(X)$
Inverse Secant	$\text{Arcsec} = \text{Atn}(X / \text{Sqr}(X * X - 1)) + \text{Sgn}(\text{Sgn}(X) - 1) * 1.5708$
Inverse Cosecant	$\text{Arccosec} = \text{Atn}(X / \text{Sqr}(X * X - 1)) + (\text{Sgn}(X) - 1) * 1.5708$
Inverse Cotangent	$\text{Arccotan} = \text{Atn}(X) + 1.5708$
Hyperbolic Secant	$\text{HSec} = 2 / (\text{Exp}(X) + \text{Exp}(-X))$
Hyperbolic Cosecant	$\text{HCosec} = 2 / (\text{Exp}(X) - \text{Exp}(-X))$
Hyperbolic Cotangent	$\text{HCotan} = (\text{Exp}(X) + \text{Exp}(-X)) / (\text{Exp}(X) - \text{Exp}(-X))$
Inverse Hyperbolic Sine	$\text{HArcsin} = \text{Log}(X + \text{Sqr}(X * X + 1))$
Inverse Hyperbolic Cosine	$\text{HArccos} = \text{Log}(X + \text{Sqr}(X * X - 1))$
Inverse Hyperbolic Tangent	$\text{HArctan} = \text{Log}((1 + X) / (1 - X)) / 2$
Inverse Hyperbolic Secant	$\text{HArcsec} = \text{Log}((\text{Sqr}(-X * X + 1) + 1) / X)$
Inverse Hyperbolic Cosecant	$\text{HArccosec} = \text{Log}((\text{Sgn}(X) * \text{Sqr}(X * X + 1) + 1) / X)$
Inverse Hyperbolic Cotangent	$\text{HArccotan} = \text{Log}((X + 1) / (X - 1)) / 2$

### 10.6.3.2 Intrinsic Functions

**ACOS**

Returns the arc cosine of a number.

Syntax

```
x = ACOS(source)
```

**ASIN**

The ASIN function returns the arc sin of a number.

Syntax

`x = ASIN(source)`

**ATN**

Returns the arctangent of a number.

Syntax

`x = ATN(source)`

**ATN2**

Returns the arctangent of  $y / x$ .

Syntax

`x = ATN(y , x)`

**COS**

Returns the cosine of an angle specified in radians.

Syntax

`x = COS(source)`

**SIN**

Returns the sine of an angle.

Syntax

`x = SIN(source)`

**TAN**

Returns the tangent of an angle.

Syntax

`x = TAN(source)`

## 10.6.4 Arithmetic Functions

**ABS**

Returns the absolute value of a number.

Syntax

`x = ABS(source)`

**EXP**

Returns  $e$  (the base of natural logarithms) raised to a power

Syntax

`x = EXP(source)`

**FRAC**

Returns the fractional part of a number.

Syntax

`x = FRAC(source)`

**INT or FIX**

Return the integer portion of a number.

Syntax

`x = INT(source)`

`x = Fix(source)`

**LOG**

Returns the natural logarithm of a number. Ln and Log perform the same function.

## Syntax

```
x = LOG(source)
x = LN(source)
```

---

**Note** LOGN = LOG(X) / LOG(N)

---

**LOG10**

The LOG10 function returns the base 10 logarithm of a number.

## Syntax

```
x = LOG10 (number)
```

**MOD**

Divides two numbers and returns only the remainder.

## Syntax

```
result = operand1 MOD operand2
```

**RectPolar**

Converts from rectangular to polar coordinates.

## Syntax

```
RectPolar (Dest, Source)
```

**SGN**

Finds the sign value of a number.

## Syntax

```
x = SGN(source)
```

**Sqr**

Returns the square root of a number.

## Syntax

```
x = SQR(number)
```

## 10.6.5 Spatial Processing

**AvgSpa**

Computes the spatial average of the values in the source array.

## Syntax

```
AvgSpa (Dest, Swath, Source)
```

**CovSpa**

Computes the spatial covariance of sets of data.

## Syntax

```
CovSpa (Dest, NumOfCov, SizeOfSets, CoreArray,
        DatArray)
```

**MaxSpa**

Finds the maximum value in an array.

## Syntax

```
MaxSpa (Dest, Swath, Source)
```

**MinSpa**

Finds the minimum value in an array.

Syntax

MinSpa (Dest, Swath, Source)

**RMSSpa**

Computes the RMS (root mean square) value of an array.

Syntax

RMSSpa (Dest, Swath, Source)

**StdDevSpa**

Used to find the standard deviation of an array.

Syntax

StdDevSpa (Dest, Swath, Source)

## 10.6.6 Other Functions

**Randomize**

Initializes the random-number generator.

Syntax

Randomize (source)

**RND**

Generates a random number.

Syntax

RND (source)

## 10.7 Clock Functions

**ClockSet**

Sets the datalogger clock from the values in an array

Syntax

ClockSet (Source)

**IfTime**

Returns a number indicating True (-1) or False (0) based on the datalogger's real-time clock.

Syntax

If (IfTime (TintoInt, Interval, Units)) Then

-or-

Variable = IfTime (TintoInt, Interval, Units)

**RealTime**

Parses year, month, day, hour, minute, second, micro-second, day of week, and/or day of year from the datalogger clock.

Syntax

RealTime (Dest)

**Ticker250ms**

Stores the running total of a 250ms system timer.

Syntax

Ticker250ms (Destination)

## 10.8 Serial Input / Output

### Print

Sends the values from program variables or other characters out through a communications port.

#### Syntax

```
Print (PrintPort, PrintBaud, PrintParams)
```

### SerialInput

Reads a serial sensor connected to the CR200(X)'s RS232 port.

#### Syntax

```
SerialInput (Dest, Max_Values, Termination_Char,  
FilterString)
```

## 10.9 Peer-to-Peer PakBus Communications

---

**Read More!** See [PakBus Overview](#) (p. 133) for more information. Also see Campbell Scientific PakBus® Networking Guide available at [www.campbellsci.com](http://www.campbellsci.com).

---

Peer-to-peer PakBus® instructions enable the datalogger to communicate with other PakBus® devices. Instructions specify a COM port and a PakBus® address. If the route to the device is not yet known, a direct route through the specified COM port is first tried.

The PakBus® Address is a variable that can be used in CRBASIC like any other variable.

The ComPort specifies the communications port that will be used to communicate with the host device. Enter one of the following commands:

- RF
- RS232

In general, PakBus® instructions write a result code to a variable indicating success or failure. Success sets the result code to 0. Otherwise, the result code increments. If communication succeeds but an error is detected, a negative result code is set. See CRBASIC Editor Help for an explanation of error codes.

The Timeout parameter in these instructions is in units of 0.01 seconds. If 0 is used, then the default timeout defined by the time of the best route is used. Use PakBusGraph "Hop Metrics" to calculate this time.

These communication instructions wait for a response or timeout before the program moves on to the next instruction.

### **GetValue**

Retrieves values from a variable in a data table of a PakBus datalogger.

#### **Syntax**

```
GetVariables (ResultCode, ComPort, NeighborAddr,  
             PakBusAddr, Security, TimeOut, "TableName",  
             "FieldName", Variable, Swath)
```

### **ReadSendGetInfo**

Returns the interval and offset for the SendGetData instruction.

#### **Syntax**

```
ReadSendGetInfo (Dest, Port)
```

### **SendData**

Sends the most recent record from a data table to a remote PakBus device.

#### **Syntax**

```
SendData (ComPort, NeighborAddr, PakBusAddr,  
          DataTable)
```

---

Note: For the CR200 series, this instruction is supported only in a special 'S' operating system, version 5.10 and greater (V06S).

---

### **SendGetData**

Sends an array of values from a remote CR200(X) to a host datalogger and/or retrieves an array of data from the host datalogger.

#### **Syntax**

```
SendGetData (ResponseDest, Control, Measurement,  
            Port, HostAddr, RepeatAddr, Security)
```

### **SetValue**

Sends values from one or more variables to a remote datalogger's Public table.

#### **Syntax**

```
SetValue (ResponseDest, Source, Swath, RemoteVar,  
         Port, RemoteAddr, RepeatAddr, Security)
```

### **TimeUntilTransmit**

The TimeUntilTransmit instruction returns the time remaining, in seconds, before communication with the host datalogger.

#### **Syntax**

```
TimeUntilTransmit (port)
```

## **10.10 Data Table Access and Management**

Commands to access and manage data stored in data tables, including Public and Status tables.

### **SetStatus**

Changes the value for a setting in the datalogger Status table.

#### **Syntax**

```
SetStatus ("FieldName", Value)
```



**TableName.FieldName**

Accesses a specific field from a record in a table

Syntax

```
TableName.FieldName (FieldNameIndex, RecordsBack)
```

**TableName.Output**

Determine if data was written to a specific DataTable the last time the DataTable was called.

Syntax

```
TableName.Output (1,1)
```

**TableName.Record**

Determines the record number of a specific DataTable record.

Syntax

```
TableName.Record (1,n)
```

**TableName.TableSize**

Returns the number of records allocated for a data table

Syntax

```
TableName.TableSize (1,1)
```

**TableName.TimeStamp**

Returns the time into an interval or a timestamp for a record in a specific DataTable.

Syntax

```
TableName.TimeStamp (m,n)
```

## 10.11 SCADA

---

**Read More!** See [Modbus](#) (p. 139).

Note: These instructions are supported only in a special operating system version 'M' for CR200 series dataloggers. That operating system must be loaded on the datalogger and the program compiled with the matching compiler to use ModBusMaster and ModBusSlave. No special operating system is required for CR200(X) series dataloggers.

---

**ModBusMaster**

Sets up a datalogger as a ModBus master to send or retrieve data from a ModBus slave.

Syntax

```
ModBusMaster (ResultCode, ComPort, BaudRate,  
              ModBusAddr, Function, Variable, Start, Length,  
              Tries, TimeOut)
```

**ModBusSlave**

Sets up a datalogger as a ModBus slave device.

Syntax

```
ModBusSlave (ComPort, BaudRate, ModBusAddr,  
             ModBusVariable, BooleanVariable, ByteOrder,  
             Offset)
```

## 10.12 Satellite Systems Programming

Instructions for GOES. Refer to satellite transmitter manuals available at [www.campbellsci.com](http://www.campbellsci.com).

### 10.12.1 GOES

The CR295 and CR295X dataloggers support communication through Campbell Scientific's TX312 or HDR GOES satellite transmitters. The CR295(X) has an extra 9-pin serial port. The CR295 requires a special operating system and does not support radio telemetry or calculation of evapotranspiration and is not CE compliant. The CR295X has the same operating system and expanded capabilities as other CR200(X) series dataloggers.

CRBASIC instructions used by the CR295(X) are:

#### **GOESData**

Sends data to a CSI GOES satellite data transmitter.

Syntax

```
GOESData (Dest, Table, TableOption, BufferControl,  
          DataFormat)
```

#### **GOESGPS**

Stores GPS data from the satellite into two variable arrays.

Syntax

```
GOESGPS (GoesArray1(6), GoesArray2(7))
```

#### **GOESSetup**

Programs the GOES transmitter for communication with the satellite.

Syntax

```
GOESSetup (ResultCode, PlatformID, MsgWindow,  
           STChannel, STBaud, RChannel, RBaud, STInterval,  
           STOffset, RInterval)
```

#### **GOESStatus**

Requests status and diagnostic information from a CSI GOES satellite transmitter.

Syntax

```
GOESStatus (Dest, StatusCommand)
```

# Section 11. Programming Resource Library

---

## 11.1 Remote Sensor Interface

The CR200(X) is frequently used as a remote sensor interface for a “Host” datalogger. Typically, the host datalogger and the sensor(s) (CR200(X)) will have programs that will enable the sensors to operate with the minimum quiescent current drain (110  $\mu$ a). These programs synchronize the CR200(X) sensors so that they are reporting back the data in designated time slots.

For applications demanding more frequent communication, the get/set variable instructions in the host datalogger can be used with CR200(X) sensors that are configured for higher current drain (250  $\mu$ a for 8 second response; 20 ma for 1 second response).

In the lowest power, synchronized mode, the CR200(X) always initiates communication. Except when it wakes up to send data, its radio is off, drawing no power. The host datalogger’s radio will be in a fully on position during the period of expected sensor communication, ready to receive and respond instantly.

CRBASIC EXAMPLE 16.	Example Wireless Sensor Program For CR200(X)
---------------------	--

<pre>Const MT = 20 Const Port = 1 Const RouterAddr = 1 Const HostAddr = 1 Const NumVals = 8 Const Security = 0 Const NumControl = 4  Public Measurements (NumVals) Public Control (NumControl)  BeginProg   Scan(1, sec)     If TimeUntilTransmit(Port)=MT Then       'applicable Measurements that take MT seconds inserted here     EndIf   NextScan EndProg</pre>	<pre>'Measurement Time (secs) SDI-12 in this Case takes 20 seconds '1 = Radio, 2 = RS-232 To send out data 'the PakBus address of a router, in this Case the same as Host 'the PakBus address of the Host datalogger '8 Measurements 'use non zero If Host datalogger has Security set  'sensor values To send 'Control values returned from Host  'If it Is time To measure start Measurement</pre>
--	--

TimeUntilTransmit(port) is a function that returns the number of seconds before it is time to communicate. It will use information retrieved from the datalogger to determine the time remaining before its communication time slot.

SendGetData will output an array of measurements to the Host datalogger and retrieve a time slot, a clock setting, and optionally an array of data from the Host

Either the output array or the input array or both can be specified as 0 meaning no data flow in the corresponding direction.

The HostAddr parameter is the PakBus Address of the master of the network, the destination of the sensor's data. The RouterAddr is the PakBus Address of a router if the distance is too far to reach the Host directly. Typically a router will not be present so this address will be identical to the HostAddr. If it is 0, then the system will automatically find a neighbor it can route through.

All communication with the host datalogger (CR800, CR1000 or CR3000) takes place through the SendGetData() instruction. When SendGetData() is included in the CR200 CRBASIC program, the CR200 calls the host approximately every 60 seconds with a random time into interval setting to establish the first connection. At first connection, the CR200 sends a variable array to the host, retrieves a variable array from the host, and retrieves a call time slot and a clock synchronization setting. Thereafter, the CR200 calls at the appointed time slot.

## 11.2 Radio Power Minimization

Power settings on the CR200(X) radio are used to optimize the current drain of the system. The radio can be set to be always on, power up once a second or every 8 seconds, or be turned on by the CR200(X) program. When the status table field RF\_ForceOn is set true under program control, the radio is on only as long as necessary for data transmission and receiving. This provides the lowest power usage by the radio.

RF\_ForceOn can be controlled with table access syntax (SetStatus(Rf\_ForceOn,1) for example) or it is controlled by the instruction SendGetData() as shown in the following [example](#) p. 111.

A CR1000 datalogger with PakBus address 2 and a RF401 radio is configured as the "master". It sends a Network() command every 5 minutes to synchronize the radio transmission. A CR206X datalogger with PakBus address 3 measures its sensors every 15 minutes and runs a countdown using TimeUntilTransmit(). When the countdown reaches zero, the CR206X powers its radio and transmits data to the CR1000 as well as receiving the CR1000 battery voltage.

The radio address and hop sequence of the CR206X must match the same settings in the RF401.

### CRBASIC EXAMPLE 17. CRBASIC EXAMPLE. Radio Power Minimization Program Examples

```
'Pin Enabled Radio Program Example Program for CR1000
'Pakbus Address = 2

Public Rx(6), Tx, Result
Alias Rx(1)=VWC_1: Alias Rx(2)=VWC_2 'Alias used to give meaningful names to
Alias Rx(3)=VWC_3: Alias Rx(4)=VWC_4 'data received from remote datalogger
Alias Rx(5)=Tsoil C: Alias Rx(6)=Batt_206_V
Alias Tx=Batt_CR1K_V

DataTable (Test,1,-1)
    DataInterval (0,60,Min,10)
    Average (6,Rx(),FP2,False)
    Minimum (1,Batt_CR1K_V,FP2,False,False)
EndTable

BeginProg
    Scan (15,Sec,0,0)
        Battery (Batt_CR1K_V)
        CallTable Test
    NextScan

'Send Network command every 5 minutes
    SlowSequence
    Scan (5,Min,0,0)
        Network (Result,1,3,15,60,0,6,Rx(),1,Tx)
    NextScan
EndProg

-----

'Pin Enabled Radio Example for CR206X
'Pakbus address 3
Public Time, TxData(6)
Public Period_uS(4), Response, Rx
Alias TxData(1)=VWC_1: Alias TxData(2)=VWC_2: Alias TxData(3)=VWC_3
Alias TxData(4)=VWC_4: Alias TxData(5)=T109_C: Alias TxData(6)=Batt_Volt
Alias Rx = CR1K_Batt_Volt

Dim LoopCnt
Units Batt_Volt=Volts
Units Period_uS()=uSec
Units T109_C=Deg C

DataTable (VWC_pin,True,-1)
    DataInterval (0,60,Min)
    Average (5,TxData(),False)
    Minimum (1,Batt_Volt,False,0)
EndTable

BeginProg
    SetStatus (RfPwrMode,RfPinEn) 'Configure CR200(X) power mode to pin enabled
    Scan(1,sec)
        If IfTime (0,15,Min) Then 'Take readings every 15 minutes
            Battery(Batt_Volt)
            Therm109(T109_C,1,5,1,1.0,0.0)
            SWBatt (1) 'Power CS625 sensors up
            PeriodAvg(Period_uS(1),1,0,100,10,1,1,0)
            PeriodAvg(Period_uS(2),2,0,100,10,1,1,0)
            PeriodAvg(Period_uS(3),3,0,100,10,2,1,0)
            PeriodAvg(Period_uS(4),4,0,100,10,2,1,0)
            SWBatt (0) 'Power CS625 sensors down
            For LoopCnt = 1 To 4 'convert raw values to volumetric water content
                TxData(LoopCnt)=-0.0663-0.0063*Period_uS(LoopCnt)+0.0007*Period_uS(
            Next LoopCnt
        EndIf

        Time = TimeUntilTransmit(1) 'copy countdown time to public variable
        If Time = 0 Then 'when countdown reaches zero power radio and transmit
            SendGetData (Response,Rx,TxData(),1,2,2,00000)
        EndIf

        CallTable (VWC_Pin)
    NextScan
EndProg
```

## 11.3 Multiple Switch Closure Measurements

Pulse channel P\_SW detects switch closures but P\_LL does not. In order to detect more than one switch closure device, a 100kOhm pull-up resistor must be connected between Battery+ and P\_LL, C1, or C2. With the pull-up resistor in place P\_LL or the control port can detect a switch closure. In the following [example](#) p. 112, note that the PulseCount instruction for P\_SW uses configuration option 2 (Switch Closure), while P\_LL must use option 0 (Pulse Input).

### CRBASIC EXAMPLE 18. CRBASIC EXAMPLE. Two Rain Gages on a CR200(X)

```
'Rain(1) is wired to P_SW and Ground
'Rain(2) is wired to P_LL and Ground with a
'100,000 ohm resistor connecting P_LL to Battery+

Public Rain(2)

Units Rain()=mm

DataTable(Hourly,True,-1)
    DataInterval(0,60,Min)
    Totalize(2,Rain(),False)
EndTable

BeginProg
    Scan(10,Sec)
        PulseCount(Rain(1),P_SW,2,0,0.254,0)
        PulseCount(Rain(2),P_LL,0,0,0.254,0)
        CallTable(Hourly)
    NextScan
EndProg
```

## 11.4 SDI-12 Sensor Support

### 11.4.1 SDI-12 Command Basics

Commands have three components:

*Sensor address (a)* – a single character, and is the first character of the command. Sensors are usually assigned a default address of zero by the manufacturer. Wildcard address (?) is used in Address Query command. Some manufacturers may allow it to be used in other commands.

*Command body (e.g., M1)* – an upper case letter (the “command”) followed by alphanumeric qualifiers.

*Command termination (!)* – an exclamation mark.

An active sensor responds to each command. Responses have several standard forms and terminate with <CR><LF> (carriage return – line feed).

SDI-12 commands and responses are defined by the SDI-12 Support Group ([www.sdi-12.org](http://www.sdi-12.org)) and are summarized in [TABLE. Standard SDI-12 Command & Response Set](#) (p. 113). Sensor manufacturers determine which commands to support. The most common commands are detailed below.

<b>Table 14. Standard SDI-12 Command &amp; Response Set</b>		
<b>Name</b>	<b>Command<sup>1</sup></b>	<b>Response<sup>2</sup></b>
Break	Continuous spacing for at least 12 milliseconds	None
Acknowledge Active	a!	a<CR><LF>
Send Identification	al!	alccccccmmmmmmvvvxxx...xx<CR><LF>
Change Address	aAb!	b<CR><LF> (support for this command is required only if the sensor supports software changeable addresses)
Address Query	?!	a<CR><LF>
Start Measurement <sup>3</sup>	aM!	atttn<CR><LF>
Start Measurement and Request CRC <sup>3</sup>	aMC!	atttn<CR><LF>
Send Data	aD0! . . . aD9!	a<values><CR><LF> or a<values><CRC><CR><LF> a<values><CR><LF> or a<values><CRC><CR><LF> a<values><CR><LF> or a<values><CRC><CR><LF> a<values><CR><LF> or a<values><CRC><CR><LF> a<values><CR><LF> or a<values><CRC><CR><LF>
Additional Measurements <sup>3</sup>	aM1! . . . aM9!	atttn<CR><LF> atttn<CR><LF> atttn<CR><LF> atttn<CR><LF> atttn<CR><LF>
Additional Measurements and Request CRC <sup>3</sup>	aMC1! ... aMC9!	atttn<CR><LF>
Start Verification <sup>3</sup>	aV!	atttn<CR><LF>
Start Concurrent Measurement	aC!	atttnn<CR><LF>
Additional Concurrent Measurements	aC1! . . . aC9!	atttnn<CR><LF> atttnn<CR><LF> atttnn<CR><LF> atttnn<CR><LF> atttnn<CR><LF>
Additional Concurrent Measurements and Request CRC	aCC1! ... aCC9!	atttnn<CR><LF>
Continuous Measurements	aR0! ... aR9!	a<values><CR><LF> (formatted like the D commands)
Continuous Measurements and Request CRC	aRC0! ... aRC9!	a<values><CRC><CR><LF> (formatted like the D commands)
<sup>1</sup> If the terminator '!' is not present, the command will not be issued. The CRBASIC SDI12Recorder () instruction, however, will still pick up data resulting from a previously issued "C!" command. <sup>2</sup> Complete response string can be obtained when using the SDIRecorder () instruction by declaring the Destination variable as String. <sup>3</sup> This command may result in a service request.		

### 11.4.1.1 Addressing

A single probe should be connected to an SDI-12 input when using these commands.

#### 11.4.1.1.1 Address Query Command (?)

Command *?!* requests the address of the connected sensor. The sensor replies to the query with the address, *a*.

#### 11.4.1.1.2 Change Address Command (aAb!)

Sensor address is changed with command *aAb!* , where *a* is the current address and *b* is the new address. For example, to change an address from 0 to 2, the command is *0A2!* . The sensor responds with the new address *b*, or in this example 2.

#### 11.4.1.1.3 Send Identification Command (a!)

Sensor identifiers are requested by issuing command *a!* . The reply is defined by the sensor manufacturer, but usually includes the sensor address, SDI-12 version, manufacturer's name, and sensor model information. Serial number or other sensor specific information may also be included.

An example of a response from the *a!*  command is:

```
013NRSYSINC1000001.2101 <CR><LF>
```

where:

- Address = 0
- SDI-12 version = 1.3
- Manufacturer = NRSYSINC
- Sensor model = 100000
- Sensor version = 1.2
- Serial number = 101

### 11.4.1.2 Start Measurement Commands (aM! & aC!)

A measurement is initiated with *M!*  or *C!*  commands. The response to each command has the form *attnn*, where

- **a** = sensor address
- **ttt** = time, in seconds, until measurement data are available
- **nn** = the number of values to be returned when one or more subsequent *D!*  commands are issued.



**11.4.1.2.1 Start Measurement Command (*aMv!*)**

Qualifier *v* is a variable between 1 and 9. If supported by the sensor manufacturer, *v* requests variant data. Variants may include:

- alternate units (e.g. °C or °F)
- additional values (e.g., level and temperature)
- diagnostic of the sensor's internal battery

Example:

Command: *5M!*

Response: 500410 (attnn, indicates address 5, data ready in 4 seconds, will report 10 values).

Example:

Command: *5M7!*

Response: 500201 (attnn indicates address 5, data ready in 2 seconds, will report 1 value). *v* = 7 instructs the sensor to return the voltage of its internal battery.

**11.4.1.2.2 Start Concurrent Measurement Command (*aC!*)**

Concurrent measurement allows the CR200(X) to request a measurement, continue program execution, and pick up the requested data on the next pass through the program. The datalogger scan rate should be set such that the resulting skew between time of measurement and time of data collection does not compromise data integrity.

---

Note This command is new to Version 1.2 or higher of the SDI-12 Specification. Older sensors, older loggers, or new sensors that do not meet v1.2 specifications will likely not support this command

---

**11.4.1.2.3 Aborting a Measurement Command**

A measurement command (*M!* or *C!*) is aborted when any other valid command is sent to the sensor.

**11.4.1.3 Send Data Commands (*aDv!* & *aRv!*)****11.4.1.3.1 Send Data Command (*aDv!*)**

This command requests data from the sensor. It is normally issued automatically by the datalogger after measurement commands *aMv!* or *aCv!*. In transparent mode, the user asserts this command to obtain data. If the expected number of data values are not returned in response to a *aD0!* command, the data

logger issues *aD1!*, *aD2!*, etc., until all data are received. The limiting constraint is that the total number of characters that can be returned to a *aDv!* command is 35 characters (75 characters for *aCv!*). If the number of characters exceed the limit, the remainder of the response are obtained with subsequent *aDv!* commands wherein *v* increments with each iteration.

#### 11.4.1.3.2 Continuous Measurements Command (*aRv!*)

Sensors designed to continuously monitor the measured phenomena, such as a shaft encoder, do not require a measurement command (e.g., *aMv!*). They are read directly with the *aRv!* command. If a sensor cannot perform continuous measurements, then it responds to *aRv!* with the sensor address.

### 11.4.2 SDI-12 Communications

The CR200(X) supports SDI-12 communication through two modes – transparent mode and programmed mode.

- Transparent mode facilitates sensor set-up and troubleshooting. It allows commands to be manually issued and the full sensor response viewed. Transparent mode does not log data.
- Programmed mode automates much of the SDI-12 protocol and provides for data recording.

Multiple SDI-12 sensors can be connected to an SDI-12 input: . If multiple sensors are wired to a single channel, each sensor must have a unique address. SDI-12 standard v 1.3 sensors accept addresses 0 - 9, a - z, and A - Z.

If program mode communications is not successful, NAN will be loaded into the first variable specified in `SDI12Recorder()` instruction. See Section [NAN and ±INF](#) (p. 153) for more information.

#### 11.4.2.1 SDI-12 Transparent Mode

System operators can manually interrogate and enter settings in probes using transparent mode. Transparent mode is useful in troubleshooting SDI-12 systems because it allows direct communication with probes.

Transparent mode may need to wait for commands issued by the programmed mode to finish before sending responses. While in transparent mode, datalogger programs may not execute. Datalogger security may need to be unlocked before transparent mode can be activated.

Transparent mode is entered while the PC is in telecommunications with the datalogger through a terminal emulator program. It is easily accessed through Campbell Scientific datalogger support software, but may also be accessible with terminal emulator programs such as Windows Hyperterminal. Keyboard displays cannot be used.

To enter the SDI-12 transparent mode, enter the support software terminal emulator. Click the “Open Terminal” button. A green “Active” indicator

appears on the screen as shown in [FIGURE. Entering SDI-12 Transparent Mode](#) (p. 117). Press <Enter> until the CR200(X) responds with the prompt “CR200(X)>”. Type “SDI12” at the prompt (without the quotes) and press <Enter>. An “Entering SDI12 Terminal” response indicates that SDI-12 Transparent Mode is active and ready to transmit SDI-12 commands and display responses. The command entered cannot exceed 12 characters. For example “0XCONFIG2=1!” is a 12 character command. This feature may limit using the extended commands offered by some sensor manufacturers, but it will not affect standard SDI-12 measurement commands and data queries.

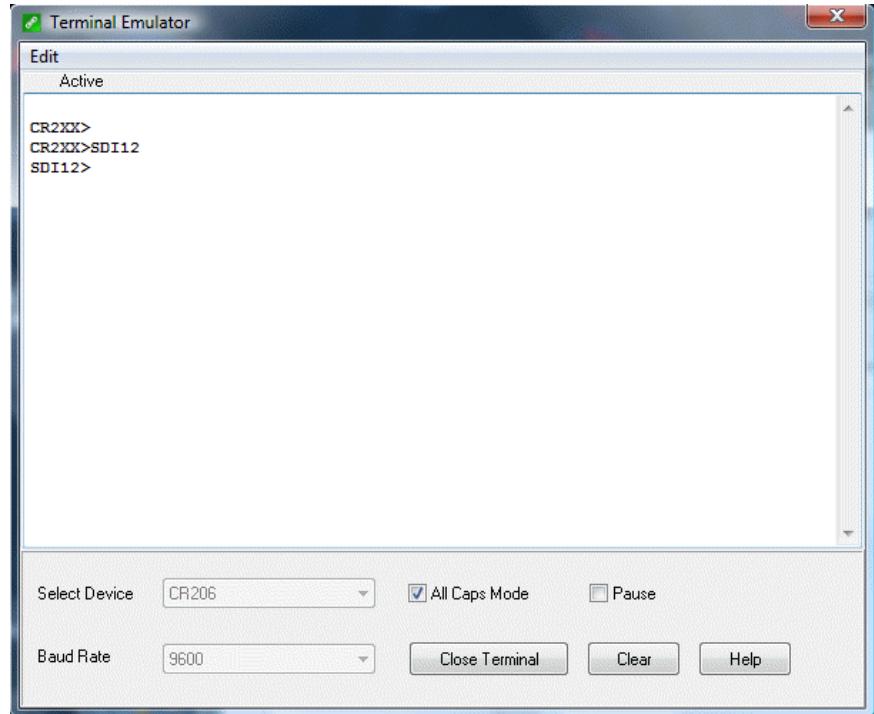


Figure 44: Entering SDI-12 Transparent Mode

#### 11.4.2.2 SDI-12 Programmed Mode

Routine SDI-12 measurements require that issuance of commands and interpretation of sensor responses be automated. Commands entered into the SDIRecorder () instruction differ slightly in function from commands entered in transparent mode. In transparent mode, for example, the operator manually enters *aM!* and *aD0!* to initiate a measurement and get data, with the operator providing the proper time delay between the request for the measurement and the request for the data. In programmed mode, the CR200(X) provides command and timing services within a single line of code. For example, when the SDI12Recorder() instruction is programmed with the M! command (note that SDI-12 address is a separate parameter), the CR200(X) issues the *aM!* AND *aD0!* commands with proper elapsed time between the two. The CR200(X) automatically issues retries as well as other services designed to make the SDI-12 measurement work as trouble free as possible. [TABLE. SDI-12Recorder \(\) Commands](#) p. 118 summarizes CR200(X) actions triggered by some SDI12Recorder commands.

<i>SDIRecorder () Instruction SDICommand Entry</i>	<i>Actions Internal to CR200(X) and Sensor</i>
<i>Mv!</i>	<b>CR200(X):</b> Issues <i>aMv!</i> command
	<b>Sensor:</b> Responds with <i>attnn</i>
	<b>CR200(X):</b> Waits until <i>ttt</i> <sup>1</sup> seconds. Issues <i>aDv!</i> command(s)
	<b>Sensor:</b> Responds with data.
<i>Cv!</i>	<b>CR200(X):</b> Issues <i>aCv!</i> command
	<b>Sensor:</b> Responds with <i>attnn</i>
	<b>CR200(X):</b> If <i>ttt</i> =0 then issues <i>aDv!</i> command(s)
	<b>Sensor:</b> Responds with data.
	<b>CR200(X):</b> else, if <i>ttt</i> >0 then moves to next CRBASIC program instruction.
	<b>CR200(X):</b> At next time SDIRecorder() is executed, if elapsed time < <i>ttt</i> , moves to next CRBASIC instruction,.
	<b>CR200(X):</b> else, issues <i>aDv!</i> command(s)
	<b>Sensor:</b> Responds with data.
	<b>CR200(X):</b> Issues <i>aCv!</i> command (to request data for next scan).
<i>Cv</i> (note: no ! termination)	<b>CR200(X):</b> Tests to see if <i>ttt</i> expired. If <i>ttt</i> not expired, loads "1e9" into first variable then moves to next CRBASIC instruction. If <i>ttt</i> expired, issues <i>aDv!</i> command(s).
	<b>Sensor:</b> Responds to <i>aDv!</i> command(s) with data, if any. If no data, loads NAN into variable.
	<b>CR200(X):</b> moves to next CRBASIC instruction (does not re-issue <i>aCv!</i> command).
<sup>1</sup> Note that <i>ttt</i> is local only to the SDIRecorder() instruction.	

### 11.4.3 SDI-12 Power Considerations

When a command is sent by the datalogger to an SDI-12 probe, all probes on the same SDI-12 port will wake up. Only the probe addressed by the datalogger

will respond, however, all other probes will remain active until the timeout period expires.

Example:

Probe: Water Content

Power Usage:

- Quiescent: 0.25 mA
- Measurement: 120 mA
- Measurement Time: 15 s
- Active: 66 mA
- Timeout: 15 s

Probes 1, 2, 3, and 4 are connected to SDI-12 / Control Port 1.

The time line in [TABLE. Example Power Usage Profile for a Network of SDI-12 Probes](#) (p. 119) shows a 35 second power usage profile example.

Table 15. Example Power Usage Profile for a Network of SDI-12 Probes								
Sec	Command	All Probes Awake	Time Out Expires	1 mA	2 mA	3mA	4mA	Total mA
1	1M!	Yes		120	66	66	66	318
2				120	66	66	66	318
•				•	•	•	•	•
•				•	•	•	•	•
•				•	•	•	•	•
14				120	66	66	66	318
15			Yes	120	66	66	66	318
16	1D0!	Yes		66	66	66	66	264
17				66	66	66	66	264
•				•	•	•	•	•
•				•	•	•	•	•
•				•	•	•	•	•
29				66	66	66	66	264
30			Yes	66	66	66	66	264
31				0.25	0.25	0.25	0.25	1
•				•	•	•	•	•
•				•	•	•	•	•
•				•	•	•	•	•
35				0.25	0.25	0.25	0.25	1

For most applications, total power usage of 318 mA for 15 seconds is not excessive, but if 16 probes were wired to the same SDI-12 port, the resulting power draw would be excessive. Spreading sensors over several SDI-12 terminals will help reduce power consumption.

## 11.5 Wind Vector

### 11.5.1 OutputOpt Parameters

In the CR200(X) WindVector () instruction, the OutputOpt parameter is used to define the values which are stored. All output options result in an array of values, the elements of which have "\_WVc(n)" as a suffix, where n is the element number. The array uses the name of the Speed/East variable as its base. [TABLE. OutputOpt Options](#) (p. 120) lists and describes OutputOpt options.

Table 16. OutputOpt Options	
Option	Description (WVc() is the Output Array)
0	WVc(1): Mean horizontal wind speed (S) WVc(2): Unit vector mean wind direction ( $\Theta_1$ ) WVc(3): Standard deviation of wind direction $\sigma(\Theta_1)$ . Standard deviation is calculated using the Yamartino algorithm. This option complies with EPA guidelines for use with straight-line Gaussian dispersion models to model plume transport.
1	WVc(1): Mean horizontal wind speed (S) WVc(2): Unit vector mean wind direction ( $\Theta_1$ )
2	WVc(1): Resultant Mean horizontal wind speed ( $\bar{U}$ ) WVc(2): Resultant mean wind direction ( $\Theta_u$ ) WVc(3): Standard deviation of wind direction $\sigma(\Theta_u)$ . This standard deviation is calculated using Campbell Scientific's wind speed weighted algorithm. Use of the resultant mean horizontal wind direction is not recommended for straight-line Gaussian dispersion models, but may be used to model transport direction in a variable-trajectory model.

### 11.5.2 Wind Vector Processing

WindVector () processes wind speed and direction measurements to calculate mean speed, mean vector magnitude, and mean vector direction over a data storage interval. Measurements from polar (wind speed and direction) or orthogonal (fixed East and North propellers) sensors are accommodated. Vector direction and standard deviation of vector direction can be calculated weighted or unweighted for wind speed.

When a wind speed measurement is zero, WindVector () uses the measurement to process scalar or resultant vector wind speed and standard deviation, but not the computation of wind direction.

---

**Note** Cup anemometers typically have a mechanical offset which is added to each measurement. A numeric offset is usually encoded in the CRBASIC program to compensate for the mechanical offset. When this is done, a measurement will equal the offset only when wind speed is zero; consequently, additional code is often included to zero the measurement when it equals the offset so that WindVector () can reject measurements when wind speed is zero.

---

### 11.5.2.1 Measured Raw Data

- $S_i$ : horizontal wind speed
- $\Theta_i$ : horizontal wind direction
- $U_{ei}$ : east-west component of wind
- $U_{ni}$ : north-south component of wind
- $N$ : number of samples

### 11.5.2.2 Calculations

#### 11.5.2.2.1 Input Sample Vectors

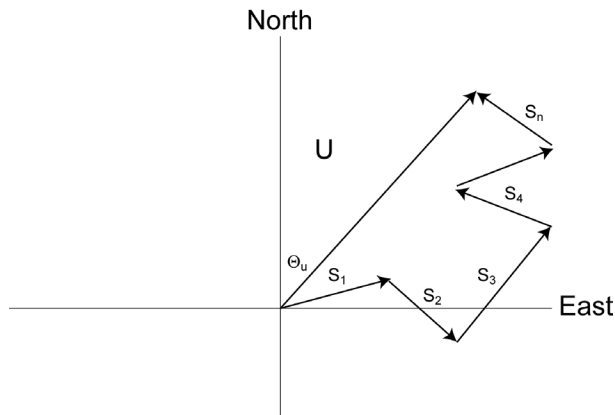


Figure 45: Input Sample Vectors

In [FIGURE. Input Sample Vectors](#) (p. 121) the short, head-to-tail vectors are the input sample vectors described by  $s_i$  and  $\Theta_i$ , the sample speed and direction, or by  $U_{ei}$  and  $U_{ni}$ , the east and north components of the sample vector. At the end of data storage interval  $T$ , the sum of the sample vectors is described by a vector of magnitude  $U$  and direction  $\Theta_u$ . If the input sample interval is  $t$ , the number of samples in data storage interval  $T$  is  $N = T/t$ . The mean vector magnitude is  $\bar{U} = U/N$ .

**Scalar mean horizontal wind speed, S:**

$$S = (\sum s_i) / N$$

where in the case of orthogonal sensors:

$$s_i = (Ue_i^2 + Un_i^2)^{1/2}$$

**Unit vector mean wind direction,**

$$\Theta_1 = \arctan (U_x / U_y)$$

where

$$U_x = (\sum \sin \Theta_i) / N$$

$$U_y = (\sum \cos \Theta_i) / N$$

or, in the case of orthogonal sensors

$$U_x = (\sum (Ue_i / U_i) / N$$

$$U_y = (\sum (Un_i / U_i) / N$$

where

$$U_i = (Ue_i^2 + Un_i^2)^{1/2}$$

**Standard deviation of wind direction (Yamartino algorithm)**

$$\sigma(\Theta_1) = \arcsin(\varepsilon)[1 + 0.1547\varepsilon^3]$$

where,

$$\varepsilon = [1 - ((U_x)^2 + (U_y)^2)]^{1/2}$$

and  $U_x$  and  $U_y$  are as defined above.



### 11.5.2.2.2 Mean Wind Vector

Resultant mean horizontal wind speed,  $\bar{U}$ :

$$\bar{U} = (U_e^2 + U_n^2)^{1/2}$$

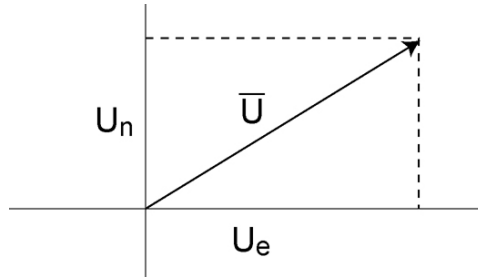


Figure 46: Mean Wind Vector

where for polar sensors:

$$U_e = (\sum s_i \sin \Theta_i) / N$$

$$U_n = (\sum s_i \cos \Theta_i) / N$$

or, in the case of orthogonal sensors:

$$U_e = (\sum U_{e_i}) / N$$

$$U_n = (\sum U_{n_i}) / N$$

Resultant mean wind direction,  $\Theta_u$ :

$$\Theta_u = \arctan (U_e / U_n)$$

Standard deviation of wind direction,  $\sigma (\Theta_u)$ , using Campbell Scientific algorithm:

$$\sigma(\Theta_u) = 81(1 - \bar{U} / S)^{1/2}$$

The algorithm for  $\sigma (\Theta_u)$  is developed by noting ([FIGURE. Standard Deviation of Direction](#) (p. 124)) that

$$\cos (\Theta_i') = U_i / s_i$$

where

$$\Theta_i' = \Theta_i - \Theta_u$$

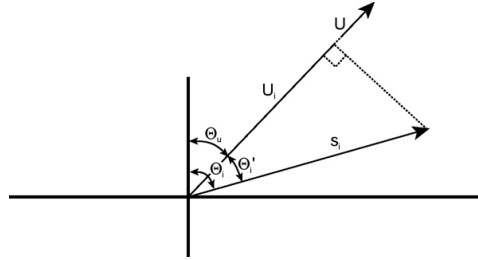


Figure 47: Standard Deviation of Direction

The Taylor Series for the Cosine function, truncated after 2 terms is:

$$\cos (\Theta_i') \cong 1 - (\Theta_i')^2 / 2$$

For deviations less than 40 degrees, the error in this approximation is less than 1%. At deviations of 60 degrees, the error is 10%.

The speed sample can be expressed as the deviation about the mean speed,

$$s_i = s_i' + S$$

Equating the two expressions for Cos ( $\theta'$ ) and using the previous equation for  $s_i$ ;

$$1 - (\Theta_i')^2 / 2 = U_i / (s_i' + S)$$

Solving for  $(\Theta_i')^2$ , one obtains;

$$(\Theta_i')^2 = 2 - 2U_i / S - (\Theta_i')^2 s_i' / S + 2s_i' / S$$

Summing  $(\Theta_i')^2$  over N samples and dividing by N yields the variance of  $\Theta_u$ . Note that the sum of the last term equals 0.

$$(\sigma(\Theta_u))^2 = (\sum_{i=1}^N (\Theta_i')^2 / N) = 2 (1 - \bar{U} / S) - \sum_{i=1}^N ((\Theta_i')^2 s_i') / NS$$

The term,

$$\sum ((\Theta_i')^2 s_i') / NS$$

is 0 if the deviations in speed are not correlated with the deviation in direction. This assumption has been verified in tests on wind data by CSI; the Air

Resources Laboratory, NOAA, Idaho Falls, ID; and MERDI, Butte, MT. In these tests, the maximum differences in

$$\sigma(\Theta_u) = (\sum(\Theta_i')^2 / N)^{1/2}$$

and

$$\sigma(\Theta_u) = (2 (1 - \bar{U} / S))^{1/2}$$

have never been greater than a few degrees.

The final form is arrived at by converting from radians to degrees (57.296 degrees/radian).

$$\sigma(\Theta_u) = (2 (1 - \bar{U} / S))^{1/2} = 81 (1 - \bar{U} / S)^{1/2}$$

## 11.6 TrigVar and DisableVar - Controlling Data Output and Output Processing

TrigVar is the third parameter in the DataTable () instruction. It controls whether or not a data record is written to final storage. TrigVar control is subject to other conditional instructions such as the DataInterval () instruction.

DisableVar is the last parameter in most output processing instructions, such as Average (), Maximum (), Minimum (), etc. It controls whether or not a particular measurement or value is included in the affected output processing function.

---

**Note** Together, TrigVar and DataInterval () grant functionality similar to Flag 0 in earlier generation CSI mixed-array dataloggers, such as the CR10X. DisableVar grants functionality similar to Flag 9.

---

For individual measurements to affect summary data, output processing instructions such as Average () must be executed whenever the DataTable is called from the program - normally once each Scan. For example, for an average to be calculated for the hour, each measurement must be added to a total over the hour. This accumulation of data are not affected by TrigVar. TrigVar only controls the moment when the final calculation is performed and the processed data (the average) is written to the data table. For this summary moment to occur, TrigVar and all other conditions (i.e. DataInterval) must be true. To restate, when TrigVar is false, output processing instructions (e.g. Average ()) perform intermediate processing but not their final processing, and a new record will not be created.

---

**Note** In many applications, output records are solely interval based and TrigVar is set to TRUE (-1) always. In these applications DataInterval () is the sole specifier of the output trigger condition.

---

*CRBASIC EXAMPLE. Using TrigVar to Trigger Data Storage* (p. 126) lists CRBASIC code that uses TrigVar () rather than DataInterval () to trigger data storage. *FIGURE. Data from TrigVar Program* (p. 126) shows data produced by the example code.

CRBASIC EXAMPLE 19.      Using TrigVar to Trigger Data Storage
In this example, the variable "counter" is incremented by 1 each scan. The data table is called Test and includes the Sample (), Average (), and Totalize () instructions. TrigVar is true when counter is equal to 0, 1, 2, 3, or 4 when the data table is called. Data are stored when TrigVar is true. Data stored are the sample, average, and total of the data table.
<pre>Public counter  DataTable (Test,counter=2 or counter=3,100)   Sample (1,counter,FP2)   Average (1,counter,FP2,False)   Totalize (1,counter,FP2,False) EndTable BeginProg   Scan (1,Sec,0,0)     counter = counter+1     If counter = 5 Then       counter = 0     EndIf     CallTable Test   NextScan EndProg</pre>

TIMESTAMP	RECORD	counter	counter	Avg	counter	Tot
TS	RN		Smp	Avg		Tot
2010-02-22 12:22:33	0	2	2	1.5	3	
2010-02-22 12:22:34	1	3	3	3	3	
2010-02-22 12:22:38	2	2	2	1.75	7	
2010-02-22 12:22:39	3	3	3	3	3	
2010-02-22 12:22:43	4	2	2	1.75	7	
2010-02-22 12:22:44	5	3	3	3	3	
2010-02-22 12:22:48	6	2	2	1.75	7	
2010-02-22 12:22:49	7	3	3	3	3	
2010-02-22 12:22:53	8	2	2	1.75	7	
2010-02-22 12:22:54	9	3	3	3	3	
2010-02-22 12:22:58	10	2	2	1.75	7	
2010-02-22 12:22:59	11	3	3	3	3	
2010-02-22 12:23:03	12	2	2	1.75	7	

Figure 48: Data from TrigVar Program

## 11.7 Multiple Data Intervals in Data Tables

The trigger variable may be used to set conditions where data is written to a data table on more than one time interval. *CRBASIC EXAMPLE. Programming for Two Data Intervals in One Data Table* p. 127 shows how this is done. Rather than using the DataInterval instruction, output times are specified by inserting If

Time instructions in the TrigVar parameter of the DataTable declaration. Since DataInterval is not used, the table size cannot be autoallocated and table size should be carefully considered before being set to a specific number of records.

#### CRBASIC EXAMPLE 20. CRBASIC EXAMPLE. Programming for two data intervals in one data table

```
'CRBASIC program to write to a single table with two different time
'intervals.

'Declare Public Variables
Public T109_C(2), Counter, deltaT
Public int_fast, int_slow
'Data Tables
'Table output on two intervals depending on condition.
'note the parenthesis around the TriggerVariable AND statements

DataTable (TwoInt,(int_fast AND IfTime(0,5,Sec)) OR (int_slow AND IfTime (0,15,sec)),1000)
    Sample (2,T109_C())
    Maximum (1,counter(1),False,False)
    Minimum (1,counter(1),False,False)
    Maximum (1,deltaT,False,False)
    Minimum (1,deltaT,False,False)
    Average (1,deltaT,false)
EndTable

'Main Program
BeginProg
    Scan (1,Sec)
        counter(1) = counter(1) + 1
        'Thermistor measurement
        Therm109 (T109_C(),2,1,Ex1,1.0,0)
        'calculate the difference in thermistor temperatures
        deltaT = T109_C(1)-T109_C(2)

        'when the difference in temperatures is >=3 turn LED on
        'and trigger the data table's faster interval
        If deltaT >= 3 Then
            PortSet (1,1)
            int_fast = -1
            int_slow = 0
        Else
            PortSet (C1,0)
            int_fast = 0
            int_slow = -1
        EndIf

        'Call Output Tables
        CallTable TwoInt
    NextScan
EndProg
```



# Section 12. Memory and Data Storage

---

## 12.1 Data Storage

The CR200(X) can be programmed to store each measurement or, more commonly, to store processed values such as averages, maxima, minima, etc. Data are stored periodically or conditionally in data tables as directed by the CRBASIC program (*CRBASIC EXAMPLE, Proper Program Structure* p. 73). The DataTable () instruction allows the user to set the size of the data table. The maximum number of tables that can be created by the program is 4 for CR200 series dataloggers and 8 for CR200(X) series dataloggers.

### 12.1.1 Data Table Storage

Data tables are stored in Serial Flash EEPROM. Data remain in memory when the CR200(X) is powered down. Data are erased when a different program is loaded and run. The Serial Flash EEPROM is good for more than 100,000 write cycles.

---

**Caution!** If an EEPROM memory failure is detected, the datalogger suspends running the program and red LED flashes twice every scan interval. “Trap Code” in the Status Table will be set to 16. The datalogger must be returned to CSI to replace the Serial Flash EEPROM.

---

The CR200(X) stores the final storage data in the Serial Flash EEPROM. The Serial Flash EEPROM also is the storage area for a file called the Table Definition File (TDF). When a CRBASIC program is downloaded to the CR200(X), the TDF is extracted from the compiled version of the CRBASIC program and stored in the Serial Flash EEPROM. Any remaining memory can be used for the final storage allocation. In the CR200, there are 128k bytes or 512k bytes of Serial Flash EEPROM (dataloggers with a 512 kbyte EEPROM will have 512K on their label). Up to 5.12k bytes of that memory can be used for the TDF (if the TDF is greater than the maximum, the compile will fail). If the data table allocation or size of all or any of the tables use too much memory, there will NOT be a compile error. The CR200(X) will force the allocation to work by adding all the sizes for each table, checking if the sum is greater than the Serial Flash EEPROM and dropping the allocation for each table by 90 percent. This is repeated until the sums of the allocations are below the Serial Flash EEPROM size.

## 12.2 Memory Conservation

One or more of the following memory saving techniques can be used on the rare occasions when a program reaches memory limits:

- Declare variables as DIM instead of Public. DIM variables do not require buffer memory for data retrieval.
- Reduce arrays to the minimum size needed. Each variable, whether or not part of an array, requires about the same amount of memory. Approximately 70 variables will fill available memory.
- Use variable arrays with aliases instead of individual variables with unique names. Aliases consume less memory than unique variable names.

## 12.3 Memory Reset

Three features are available for complete or selective reset of CR200(X) memory.

### 12.3.1 Full Memory Reset

Full memory reset occurs when an operating system is sent to the CR200(X) using DevConfig. A full memory reset does the following:

- Clears and Formats CPU: drive (all program files erased)
- Clears SRAM data tables
- Clears Status Table Elements
- Restores settings to default
- Initializes system variables
- Clears communications memory

### 12.3.2 Program Send Reset

All SRAM and CRD: data are erased when user programs are uploaded. This will occur even when a program with identical data tables is sent to the CR200(X).

### 12.3.3 Manual Data Table Reset

Data table memory is reset using the ResetTables field of the Status table. Change the value from 0 to 8888 to reset all data tables.



## Section 13. Telecommunications and Data Retrieval

---

Telecommunications, in the context of CR200(X) operation, is the movement of information between the CR200(X) and another computing device, usually a PC. The information can be programs, data, files, or control commands.

Telecommunications systems require three principal components: hardware, carrier signal, and protocol. For example, a common way to communicate with the CR200(X) is with PC200W software by way of a PC COM port. In this example, hardware are the PC COM port, the CR200(X) RS-232 port, and a serial cable. The carrier signal is RS-232, and the protocol is PakBus®. Of these three, a user most often must come to terms with only the hardware, since the carrier signal and protocol are transparent in most applications.

Systems usually require a single type of hardware and carrier signal. Some applications, however, require hybrid systems, which utilize two or more hardware and signal carriers.

Contact a Campbell Scientific applications engineer for assistance in configuring any telecommunications system.

Synopses of software to support the various telecommunications devices and protocols are found in Section [Support Software](#) (p. 143).

### 13.1 Hardware and Carrier Signal

Campbell Scientific supplies or recommends a wide range of telecommunications hardware. [TABLE. CR200\(X\) Telecommunications Options](#) (p. 131) lists telecommunications destination, device, path, and carrier options which imply certain types of hardware for use with the CR200(X) datalogger. Information in [TABLE. CR200\(X\) Telecommunications Options](#) (p. 131) is conceptual. For specific model numbers and specifications, contact a Campbell Scientific applications engineer or go to [www.campbellsci.com](http://www.campbellsci.com).

Table 17. CR200(X) Telecommunications Options		
<b>Destination Device / Portal</b>	<b>Communications Path</b>	<b>Carrier Signal</b>
PC / COM or USB	Direct Connect	RS-232
PDA / COM Port	Direct Connect	RS-232
PC / COM Port	Digital Cellular	800 MHz RF
PC / COM Port	Multidrop	RS485
PC / Network Card	Ethernet / PPP	IP
PC / COM Port	Spread Spectrum RF	900 MHz RF
Satellite System	Satellite Transceiver	RF
Digital Display	Direct Connect	RS-232

---

NOTE: The CR200(X) operates at a baud rate of 9600 baud. Attempting to connect at a higher baud rate will result in communications errors.

---

## 13.2 Protocols

The primary telecommunication protocol for the CR200(X) is PakBus ([PakBus Overview](#) (p. 133)). ModBus is also supported on board ([Alternate Telecoms Resource Library](#) (p. 139)).

## 13.3 Initiating Telecommunications

Telecommunications sessions are initiated by a PC running Campbell Scientific datalogger support software ([Support Software](#) p. 143). Once telecommunication is established, the PC issues commands to send programs, set clocks, collect data, etc. Because data retrieval is managed by the PC, several PCs can have access to a CR200(X) without disrupting the continuity of data.

## 13.4 Data Retrieval

Data tables are transferred to PC files through a telecommunications link ([Telecommunications and Data Retrieval](#) (p. 131)).

### 13.4.1 Via Telecommunications

Data are usually transferred through a telecommunications link to an ASCII file on the supporting PC using Campbell Scientific datalogger support software ([Support Software](#) (p. 143)). See also the manual and Help for the software package being used.

### 13.4.2 Data Format on Computer

CR200(X) data stored on a PC via support software is formatted as either ASCII or Binary depending on the file type selected in the support software. Consult the software manual for details on the various available data file formats.

# Section 14. PakBus Overview

---

---

**Read More!** This section is provided as a primer to PakBus® communications. Complete information is available in Campbell Scientific's "PakBus Networking Guide", available at [www.campbellsci.com](http://www.campbellsci.com).

---

The CR200(X) communicates with computers or other Campbell Scientific dataloggers via PakBus®. PakBus® is a proprietary telecommunications protocol similar in concept to IP (Internet protocol). PakBus® allows compatible Campbell Scientific dataloggers and telecommunications peripherals to seamlessly link to a PakBus® network.

## 14.1 PakBus Addresses

CR200(X)s are assigned PakBus® address 1 as a factory default. Networks with more than a few stations should be organized with an addressing scheme that guarantees unique addresses for all nodes. One approach, demonstrated in [FIGURE. PakBus Network Addressing](#) (p. 134), is to assign single-digit addresses to the first tier of nodes, multiples of tens to the second tier, multiples of 100s to the third, etc. Note that each node on a branch starts with the same digit. Devices, such as PCs, with addresses greater than 4000 are given special administrative access to the network

PakBus addresses are set using DevConfig, PakBusGraph, or the CR200(X) status table. DevConfig (Device Configuration Utility) is the primary settings editor for Campbell Scientific equipment. It requires a hardwire RS-232 connection to a PC and allows backup of settings on the PC hard drive. PakBusGraph is used over a telecommunications link to change settings, but has no provision for backup.

---

**Caution** Care should be taken when changing PakBus® addresses with PakBusGraph or in the status table. If an address is changed to an unknown value, a field visit with a laptop and DevConfig may be required to discover the unknown address.

---

## 14.2 Nodes: Leaf Nodes and Routers

- A PakBus® network consists of 2 to 4093 linked nodes.
- One or more leaf nodes and routers can exist in a network.
- Leaf nodes are measurement devices at the end of a branch of the PakBus® web.
- Leaf nodes can be linked to any router.
- A leaf node cannot route packets but can originate or receive them.

- Routers are measurement or telecommunications devices that route packets to other linked routers or leaf nodes.
- Routers can be branch routers. Branch routers only know as neighbors central routers, routers in route to central routers, and routers one level outward in the network.
- Routers can be central routers. Central routers know the entire network. A PC running LoggerNet is typically a central router.
- Routers can be router-capable dataloggers or communications devices.

The CR200(X) is always a leaf node. It cannot be configured as a router. The network shown in [FIGURE. PakBus Network Addressing](#) (p. 134) contains 6 routers and 8 leaf nodes.

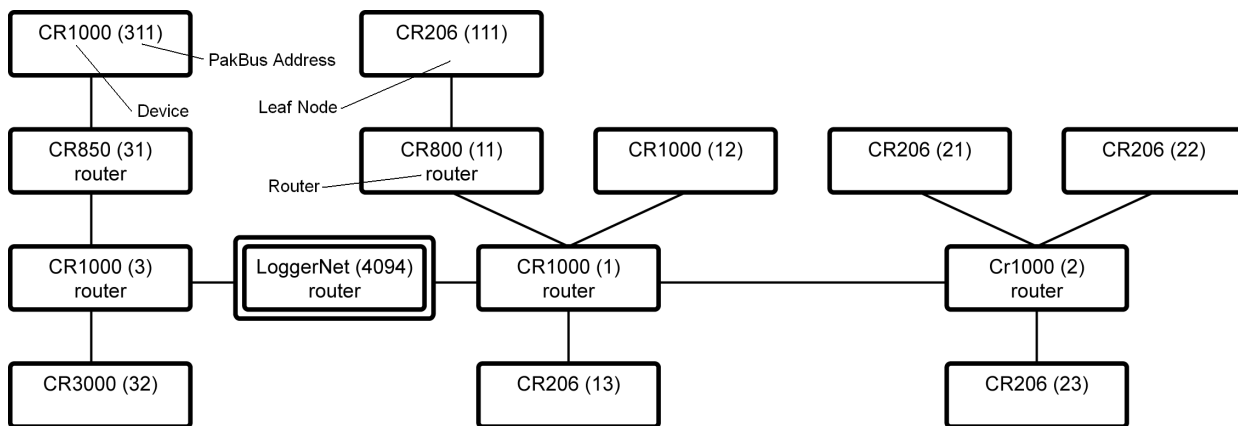


Figure 49: PakBus Network Addressing.

LoggerNet is configured by default as a router and can route datalogger to datalogger communications.

## 14.3 Router and Leaf Node Configuration

## 14.4 Linking Nodes: Neighbor Discovery

To form a network, *nodes* must establish *links* with *neighbors* (adjacent nodes). *Links* are established through a process called *discovery*. *Discovery* occurs when *nodes* exchange *hellos*. A hello exchange occurs during a *hello-message* between two nodes.

### 14.4.1 Hello-message (two-way exchange)

A *hello-message* is an interchange between two nodes that negotiates a neighbor link. A *hello-message* is sent out in response to one or both of either a beacon or a hello-request.

### 14.4.2 Beacon (one-way broadcast)

A *beacon* is a broadcast sent by a node at a specified interval telling all nodes within hearing that a *hello-message* can be sent. If a node wishes to establish itself as a neighbor to the beaconing node, it will then send a *hello-message* to the beaconing node. Nodes already established as neighbors will not respond to a beacon.

### 14.4.3 Hello-request (one-way broadcast)

All nodes hearing a *hello-request* broadcast (existing and potential neighbors) will issue a *hello-message* to negotiate or re-negotiate a neighbor relationship with the broadcasting node.

### 14.4.4 Neighbor Lists

PakBus® devices in a network can be configured with a neighbor list. The router or master sends out a *hello-message* to each node in the list whose verify interval has expired at a random interval\*. If a node responds, a hello-message is exchanged and the node becomes a neighbor.

\*A random number of seconds between INTERVAL and (2 \* INTERVAL), where INTERVAL is the Verify Interval setting if non-zero, or 30 seconds if the Verify Interval setting is zero.

Neighbor filters dictate which nodes are neighbors and force packets to take routes specified by the network administrator. LoggerNet (a PakBus® node) derives its neighbor filter from link information in the Setup device map.

### 14.4.5 Adjusting Links

PakBusGraph, a client of LoggerNet, is particularly useful when testing and adjusting PakBus® routes.

Paths established by way of beaconing may be redundant and vary in reliability. Redundant paths can provide backup links in the event the primary path fails. Redundant and unreliable paths can be eliminated by activating neighbor filters in the various nodes and by disabling some beacons.

### 14.4.6 Maintaining Links

Links are maintained by means of the CVI (communications verification interval). The CVI can be specified in each node with DevConfig. The *following rules<sup>1</sup>* apply:

- If Verify Interval = 0, then CVI = 2.5 x beacon interval\*
- If Verify Interval = 60, then CVI = 60 seconds\*
- If Beacon Interval = 0 and Verify Interval = 0, then CVI = 300 seconds\*
- If a router or master does not hear from a neighbor for one CVI, it begins again to send a Hello message to that node at the random interval.

Users should base verification intervals on the timing of normal communications such as scheduled LoggerNet collections or datalogger to dataloggers communications. The idea is to not allow the verification interval to expire before normal communications. If the verification interval expires the devices will initiate hello exchanges in an attempt to regain neighbor status, increasing traffic in the network.

---

**Note:** During the hello-message, a CVI must be negotiated between two neighbors. The negotiated CVI is the lesser of the first nodes CVI and 6/5ths of the neighbors CVI.

---

## 14.5 Troubleshooting

Various tools and methods have been developed to assist in troubleshooting PakBus® networks.

### 14.5.1 Link Integrity

With beaconing or neighbor filter discovery, links are established and verified using relatively small data packets (Hello messages). When links are used for regular telecommunications, however, longer messages are used. Consequently, a link may be reliable enough for *discovery* but unreliable with larger packets. This condition is most common in radio networks, particularly when max packet size is >200.

PakBus® communications over marginal links can often be improved by reducing the size of the PakBus® packets. Best results are obtained when the maximum packet sizes in both nodes are reduced.

---

<sup>1</sup> During the hello-message, a CVI must be negotiated between two neighbors. The negotiated CVI is the lesser of the first nodes CVI and 6/5ths of the neighbors CVI.

### 14.5.1.1 Automatic Packet Size Adjustment

The BMP5 file receive transaction allows the BMP5 client (LoggerNet) to specify the size of the next fragment of the file that the CR200(X) sends.

---

**Note** The file receive transaction is used to get table definitions from the datalogger.

---

Because LoggerNet must specify a size for the next fragment of the file, it uses whatever size restrictions that apply to the link.

Hence, the size of the responses to the file receive commands that the CR200(X) sends is governed by the maxPacketSize setting for the datalogger as well as that of any of its parents in LoggerNet's network map. Note that this calculation also takes into account the error rate for devices in the link.

BMP5 data collection transaction does not provide any way for the client to specify a cap on the size of the response message. This is the main reason why the "Max Packet Size" setting exists in the CR200(X). The CR200(X) can look at this setting at the point where it is forming a response message and cut short the amount of data that it would normally send if the setting limits the message size.

### 14.5.2 Ping

Link integrity can be verified with the following procedure by using PakBusGraph | Ping Node. Nodes can be pinged with packets of 50, 100, 200 or 500 bytes.

---

**Note** Do not use packet sizes greater than 90 when pinged with 100 mW radio modems and radio enabled dataloggers.

---

Pinging with ten repetitions of each packet size will characterize the link. Before pinging, all other network traffic (scheduled data collections, clock checks, etc.) should be temporarily disabled. Begin by pinging the first layer of links (neighbors) from the PC, then proceed to nodes that are more than one hop away. [TABLE. PakBus Link Performance Gage](#) (p. 137) provides a link performance gage.

<b>Table 18. PakBus Link Performance Gage</b>		
<b>500 byte Ping Sent</b>	<b>Successes</b>	<b>Link Status</b>
10	10	excellent
10	9	good
10	7-8	adequate
10	<7	marginal

### 14.5.3 Traffic Flow

Keep beacon intervals as long as possible with higher traffic (large numbers of nodes and / or frequent data collection). Long beacon intervals minimize collisions with other packets and resulting retries. The minimum recommended beacon interval is 60 seconds. If communications traffic is high, consider setting beacon intervals of several minutes. If data throughput needs are great, maximize data bandwidth by creating some branch routers, and / or by eliminating beacons altogether and setting up neighbor filters.

## 14.6 LoggerNet Device Map Configuration

As shown in [FIGURE. Flat Map](#) (p. 138) and [FIGURE. Tree Map](#) (p. 138), the essential element of a PakBus® network device map in LoggerNet is the PakBusPort. After adding the root port (COM, IP, etc), add a PakBusPort and the dataloggers.

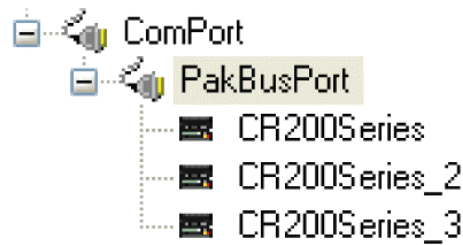


Figure 50: Flat Map

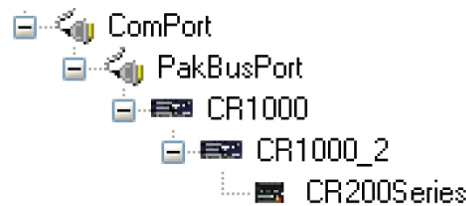


Figure 51: Tree Map

Use the 'tree' configuration when communications requires routers. The shape of the map serves to disallow a direct LoggerNet connection to CR1000\_2 and CR200Series, and implies constrained routes that will probably be established by user-installed neighbor filters in the routers. This assumes that LoggerNet beacons are turned off. Otherwise, with a default address of 4094, LoggerNet beacons will penetrate the neighbor filter of any in-range node.



# Section 15. Alternate Telecoms Resource Library

---

## 15.1 Modbus

### 15.1.1 Overview

Modbus is a widely used SCADA communication protocol that facilitates exchange of information and data between computers / HMI software, instruments (RTUs) and Modbus compatible sensors. The CR200(X) communicates via Modbus over RF and RS-232.

Modbus systems consist of a master (PC), RTU / PLC slaves, field instruments (sensors), and the communications network hardware. The communications port, baud rate, data bits, stop bits, and parity are set in the Modbus driver of the master and / or the slaves. The Modbus standard has two communications modes, RTU and ASCII. However, CR200(X)s communicate in RTU mode exclusively.

Field instruments can be queried by the CR200(X). Because Modbus has a set command structure, programming the CR200(X) to get data from field instruments is much simpler than from serial sensors. Because Modbus uses a common bus and addresses each node, field instruments are effectively multiplexed to a CR200(X) without additional hardware.

A CR200(X) goes into sleep mode after 40 seconds of communications inactivity. Once asleep, two packets are required before the CR200(X) will respond. The first packet awakens the CR200(X); the second packet is received as data.

### 15.1.2 Terminology

*TABLE. Modbus to Campbell Scientific Equivalents* (p. 139) lists terminology equivalents to aid in understanding how CR200(X)s fit into a SCADA system.

Table 19. Modbus to Campbell Scientific Equivalents		
<b>Modbus Domain</b>	<b>Data Form</b>	<b>Campbell Scientific Domain</b>
Coils	Single Bit	Ports, Flags, Boolean Variables
Digital Registers	16-bit Word	Floating Point Variables
Input Registers	16-bit Word	Floating Point Variables
Holding Registers	16-bit Word	Floating Point Variables
RTU / PLC		CR200(X)
Master		Usually a computer
Slave		Usually a CR200(X)
Field Instrument		Sensor

### 15.1.2.1 Glossary of Terms

#### Coils (00001 to 09999)

Originally, "coils" referred to relay coils. In CR200(X)s, coils are exclusively ports, flags, or a Boolean variable array. Ports are inferred if parameter 5 of the ModbusSlave instruction is set to 0. Coils are assigned to Modbus registers 00001 to 09999.

#### Digital Registers 10001-19999

Hold values resulting from a digital measurement. Digital registers in the Modbus domain are read only. In the CSI domain, the leading digit in Modbus registers is ignored, and so are assigned together to a single Dim or Public variable array (read / write).

#### Input Registers 30001 - 39999

Hold values resulting from an analog measurement. Input registers in the Modbus domain are read only. In the CSI domain, the leading digit in Modbus registers is ignored, and so are assigned together to a single Dim or Public variable array (read / write).

#### Holding Registers 40001 - 49999

Hold values resulting from a programming action. Holding registers in the Modbus domain are read / write. In the CSI domain, the leading digit in Modbus registers is ignored, and so are assigned together to a single Dim or Public variable array (read / write).

#### RTU / PLC

Remote Telemetry Units (RTUs) and Programmable Logic Controllers (PLCs) were at one time used in exclusive applications. As technology increases, however, the distinction between RTUs and PLCs becomes more blurred. A CR200(X) fits both RTU and PLC definitions.

## 15.1.3 Programming for Modbus

### 15.1.3.1 Declarations

*TABLE. CRBASIC Ports, Flags, Variables and Modbus Registers* (p. 141) shows the linkage between CR200(X) ports, flags and Boolean variables and Modbus registers. Modbus does not distinguish between CR200(X) ports, flags, or Boolean variables. By declaring only ports, or flags, or Boolean variables, the declared feature is addressed by default. A typical CRBASIC program for a Modbus application will declare variables and ports, or variables and flags, or variables and Boolean variables.

<b>Table 20. CRBASIC Ports, Flags, Variables and Modbus Registers</b>		
<b>CR200(X) Feature</b>	<b>Example CRBASIC Declaration</b>	<b>Equivalent Example Modbus Register</b>
Control Port (Port)	Public Port(8)	00001 to 00009
Flag	Public Flag(17)	00001 to 00018
Boolean Variable	Public ArrayB(56) as Boolean	00001 to 00057
Variable	Public ArrayV(20)*	40001 to 40041* <b>or</b> 30001 to 30041*
*Because of byte number differences, each CR200(X) domain variable translates to two Modbus domain input / holding registers.		

### 15.1.3.2 CRBASIC Instructions - Modbus

Complete descriptions and options of commands are available in CRBASIC Editor Help.

#### **ModbusMaster ()**

Sets up a CR200(X) as a Modbus master to send or retrieve data from a Modbus slave.

##### Syntax

```
ModbusMaster (ResultCode, ComPort, BaudRate,
              ModbusAddr, Function, Variable, Start, Length,
              Tries, TimeOut)
```

#### **ModbusSlave ()**

Sets up a CR200(X) as a Modbus slave device.

##### Syntax

```
ModbusSlave (ComPort, BaudRate, ModbusAddr,
             DataVariable, BooleanVariable)
```

### 15.1.3.3 Addressing (ModbusAddr)

Modbus devices have a unique address in each network. Addresses range from 1 to 247. Address 0 is reserved for universal broadcasts. When using the NL100, use the same number as the Modbus and PakBus® address.

### 15.1.3.4 Supported Function Codes (Function)

Modbus protocol has many function codes. CR200(X) commands support the following.

- 01 Read Coil Status
- 02 Read Input Status
- 03 Read Holding Registers
- 04 Read Input Registers

- 05 Force Single Coil
- 15 Force Multiple Coils
- 16 Force Multiple Registers

#### 15.1.3.5 Reading Inverse Format Registers

Some Modbus devices require reverse byte order words (CDAB vs. ABCD). This can be true for either floating point, or integer formats. While other CRBasic dataloggers, such as the CR1000, can use the MoveBytes() instruction to output reverse byte order words, the CR200(X) does not have that capability. If reverse byte order words are required, either the master has to make an adjustment, which is sometimes possible, or the CR200(X) needs to be replaced with a datalogger that supports the MoveBytes() instruction.

#### 15.1.4 Troubleshooting

Test Modbus functions on the CR200(X) with third party Modbus software. Further information is available at the following links:

- [www.simplymodbus.ca/FAQ.htm](http://www.simplymodbus.ca/FAQ.htm) (<http://www.simplymodbus.ca/faq.htm>)
- [www.Modbus.org/tech.php](http://www.modbus.org/tech.php) (<http://www.modbus.org/tech.php>)
- [www.lammertbies.nl/comm/info/modbus.html](http://www.lammertbies.nl/comm/info/modbus.html)  
(<http://www.lammertbies.nl/comm/info/modbus.html>)

## **Section 16. Support Software**

---

PC / Windows® compatible software products are available from Campbell Scientific to facilitate CR200(X) programming, maintenance, data retrieval, and data presentation. Short Cut, PC200W, and Visual Weather are designed for novice integrators, but have features useful in advanced applications. PC400 and LoggerNet provide increasing levels of power required for advanced integration, programming and networking applications. Support software for PDA and Linux applications are also available.

A recent addition to the support software line-up is the Campbell Scientific Network Planner, which assists in the configuration of networks.

### **16.1 Short Cut**

Short Cut utilizes an intuitive user interface to create CR200(X) program code for common measurement applications. It presents lists from which sensors, engineering units, and data output formats are selected. It supports by name most sensors sold by Campbell Scientific. It features "generic" measurement routines, enabling it to support many sensors from other manufacturers. Programs created by Short Cut are automatically well documented and produce examples of CRBASIC programming that can be used as source or reference code for more complex programs edited with CRBASIC Editor.

Short Cut is included with PC200W, Visual Weather, PC400, RTDAQ, and LoggerNet and is available at no charge from the Campbell Scientific web site.

### **16.2 PC200W**

PC200W utilizes an intuitive user interface to support direct serial communication to the CR200(X) via COM / RS-232 ports. It sends programs, collects data, and facilitates monitoring of digital measurement and process values. PC200W is available at no charge from the Campbell Scientific web site.

### **16.3 Visual Weather**

Visual Weather supports weather stations. It is recommended in applications wherein the user requires minimal control over programming and the pre-configured display and reporting features. Visual Weather is highly integrated and easy to use.

## 16.4 PC400

PC400 is a mid-level software suite. It includes CRBASIC Editor, point-to-point communications over several communications protocols, simple real-time digital and graphical monitors, and report generation. It does not support scheduled collection or multi-mode communication networks.

## 16.5 RTDAQ

RTDAQ is targeted for industrial and other high-speed data acquisition applications.

## 16.6 LoggerNet Suite

The LoggerNet suite utilizes a client-server architecture that facilitates a wide range of applications and enables tailoring software acquisition to specific requirements. [TABLE. LoggerNet Products that Include the LoggerNet Server](#) (p. 144) lists features of LoggerNet products that include the LoggerNet server. [TABLE. LoggerNet Clients](#) (p. 145) lists features of LoggerNet products that require the LoggerNet server as an additional purchase.

<b>Table 21. LoggerNet Products that Include the LoggerNet Server</b>	
LoggerNet	Datalogger management, programming, data collection, scheduled data collection, network monitoring and troubleshooting, graphical data displays, automated tasks, data viewing and post-processing.
LoggerNet Admin	All LoggerNet features plus network security, manages the server from a remote PC, runs LoggerNet as a service, exports data to third party applications, launches multiple instances of the same client, e.g., two or more functioning Connect windows.
LoggerNet Remote	Allows management of an existing LoggerNet datalogger network from a remote location, without investing in another complete copy of LoggerNet Admin.
LoggerNet-SDK	Allows software developers to create custom client applications that communicate through a LoggerNet server with any datalogger supported by LoggerNet. Requires LoggerNet.
LoggerNet Server - SDK	Allows software developers to create custom client applications that communicate through a LoggerNet server with any datalogger supported by LoggerNet. Includes the complete LoggerNet Server DLL, which can be distributed with the custom client applications.
LoggerNet Linux	Includes LoggerNet Server for use in a Linux environments and LoggerNet Remote for managing the server from a Windows environment.

<b>Table 22. LoggerNet Clients</b>	
<b><i>These LoggerNet clients require, but are not sold with, the LoggerNet Server.</i></b>	
Baler	Handles data for third-party application feeds.
RTMCRT	RTMC viewer only.
RTMC Web Server	Converts RTMC graphics to HTML.
RTMC Pro	Enhanced version of RTMC.
LoggerNetData	Displays / Processes real-time and historical data.
CSI OPC Server	Feeds data into third-party OPC applications.

## 16.7 PDA Software

PConnect Software supports PDAs with Palm Operating Systems. PConnectCE supports Windows Mobile and Pocket PC PDAs. Both support direct RS-232 connection to the CR200(X) for sending programs, collecting data, and digital real-time monitoring.

## 16.8 Network Planner

Network Planner is a "drag and drop" application used in designing PakBus datalogger networks. The user interacts with Network Planner through a drawing canvas upon which are placed PCs and dataloggers nodes. Links representing various telecommunications options are drawn between nodes. Activities to take place between the nodes are specified. Network Planner automatically specifies settings for individual devices and creates XML files for download to each device through DevConfig.





## **Section 17. Care and Maintenance**

---

Temperature and humidity can affect the performance of the CR200(X). The internal lithium battery must be replaced periodically. Factory replacement is recommended. Contact Campbell Scientific to obtain an RMA prior to shipping the CR200(X).

### **17.1 Temperature Range**

The CR200(X) is designed to operate reliably from -40°C to +50°C in non-condensing environments.

### **17.2 Moisture Protection**

When humidity tolerances are exceeded and condensation occurs, damage to IC chips, microprocessor failure, and/or measurement inaccuracies due to condensation on the various PC board runners may result. Effective humidity control is the responsibility of the user.

Adequate desiccant should be placed in the instrumentation enclosure to prevent corrosion on the CR200(X) wiring panel. The desiccant needs to be replaced or dried out frequently enough to control the humidity.

### **17.3 Enclosures**

Campbell Scientific offers environmental enclosures for housing a CR200(X) and peripherals. These enclosures are classified as NEMA 4X (watertight, dust-tight, corrosion-resistant, indoor and outdoor use).

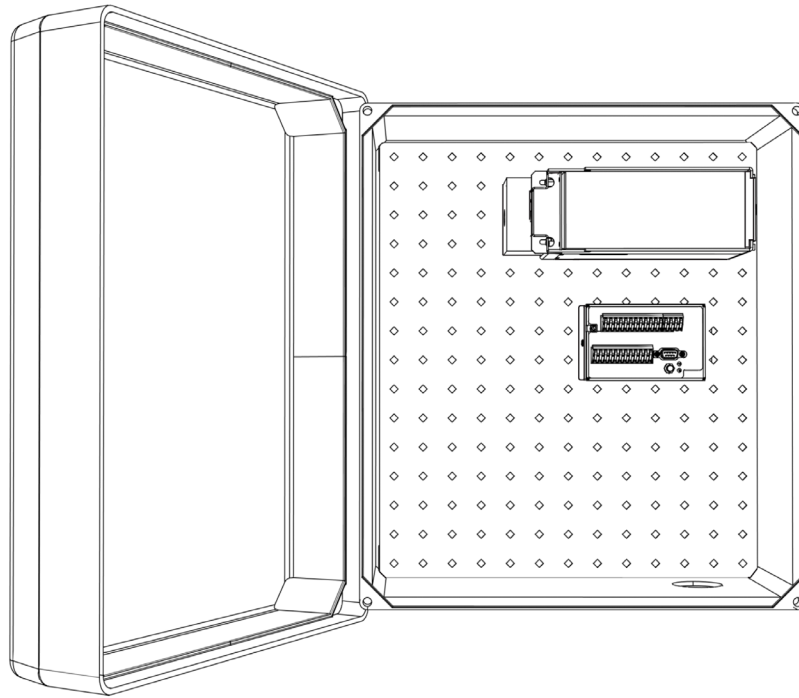


Figure 52: Enclosure

## 17.4 Replacing the Internal Battery

---

**Caution** Fire, explosion, and severe burn hazard! Misuse or improper installation of the lithium battery can cause severe injury. Do not recharge, disassemble, heat above 100°C (212°F), solder directly to the cell, incinerate, or expose contents to water. Dispose of spent lithium batteries properly.

---

The CR200(X) contains a lithium battery that operates the clock and SRAM when the CR200(X) is not powered. The CR200(X) does not draw power from the lithium battery while it is powered by an external source. In a CR200(X) stored at room temperature, the lithium battery should last approximately 5 years (less at temperature extremes). Where the CR200(X) is powered most or all of the time the lithium cell should last much longer.

The internal lithium battery must be replaced periodically. Factory replacement is recommended. Contact Campbell Scientific to obtain an RMA prior to shipping the CR200(X).

If the lithium cell is removed or allowed to discharge below the safe level, the CR200(X) will still operate correctly while powered. Without the lithium battery, the clock will reset and data is lost when power is removed.

A replacement lithium battery can be purchased from Campbell (part number 15598). [TABLE. CR200\(X\) Lithium Battery Specifications](#) p. 33 lists the specifications of the battery. However, Campbell Scientific recommends that the battery be replaced at the factory.

<b>Table 23. Internal Lithium Battery Specifications</b>	
Manufacturer	Renata
Model	CR2016 (3.6V)
Capacity	80 mAh
Self-discharge rate	1%/year @ 23°C
Operating temperature range	-40°C to +85°C



# Section 18. Troubleshooting

---

---

**Note** If any component needs to be returned to the factory for repair or recalibration, remember that an RMA number is required. Contact a Campbell Scientific applications engineer to receive the RMA number.

---

## 18.1 Programming

A properly deployed CR200(X) measures sensors accurately and stores all data as instructed by its program. Experienced users analyze data soon after deployment to ensure the CR200(X) is measuring and storing data as intended. Most measurement and data storage problems are a result of one or more instances of improper program code or "bugs."

### 18.1.1.1 Status Table as Debug Resource

Consult the CR200(X) Status Table when developing a program or when a problem with a program is suspected.

---

**Read More!** See [APPENDIX. Status Table and Settings](#) p. 15 for a complete list of Status Table registers and hints on using the Status Table.

---

### 18.1.1.2 SkippedScan

Skipped scans are caused by long programs with short scan intervals or when other operations occupy the processor at a scan's start time. Occasional skipped scans may be acceptable but should be avoided. Skipped scans may compromise frequency measurements made with pulse channels. The error occurs because counts from a scan and subsequent skipped scans are regarded by the CR200(X) as having occurred during a single scan. The measured frequency can be much higher than actual. Be careful that scans that store data are not skipped. If any scan skips repeatedly, optimization of the datalogger program or reduction of online processing may be necessary.

### 18.1.1.3 VarOutOfBounds

Indicates the number of variables that are out of bounds, meaning the dimensioned array is not large enough to store the variable. When this value is non-zero, edit the program and increase the array size.

### 18.1.1.4 WatchdogErrors

Non-zero indicates the CR200(X) has crashed, which can be caused by power or transient voltage problems, or an operating system or hardware problem. Watchdog errors may cause telecommunications disruptions, which can make diagnosis and remediation difficult. Sometimes a TrapCode will accompany a WatchDogCnt.

### 18.1.1.5 TrapCode

Normally this value is zero. If set to a value of 16, TrapCode indicates an EEPROM memory failure. When this occurs the datalogger stops running its program and the red LED flashes twice per scan interval. The datalogger must be returned to CSI to replace the Serial Flash EEPROM. Contact a Campbell Scientific applications engineer to receive an RMA number.

### 18.1.1.6 Compile and Download Errors

When a user program is compiled, it is checked for errors. Errors caught by the compiler are termed "Compile Errors." Because CR200(X) programs are compiled externally by the datalogger support software, most programming errors are found before the program reaches the CR200(X). A program wherein no compile errors are found may still have errors only detectable after the program reaches the CR200(X). These errors are "Download Errors." [TABLE. Program Download Errors](#) p. 152 lists possible error messages and conditions.

Table 24. Program Download Errors		
Download Error / Condition	Error Meaning	Notes
"No Prog Running"	Flash memory holding the program has been erased.	
"Wrong OS Ver"	Program was compiled in the CRBASIC Editor with an incorrect (usually older) compiler executable that does not match the operating system resident in the CR200(X).	<p>Compiler executable file (C:\Campbellsci\Lib\ CR200Compilers\cr2compvxxxx.exe) must match the CR200(X) operating system (cr2osvxxxx.a43), where xxxx is the operating system version number. Compiler executable version is shown on the first line of the compile results in CRBASIC Editor.</p> <p>This error typically occurs only when forcing a .BIN file to be sent to the CR200(X). By default, datalogger support software "sends" the .CR2 file, automatically matching OS and compiler.</p> <p>Current compilers can be obtained at <a href="http://www.campbellsci.com">www.campbellsci.com</a>.</p>
"Wrong Prog"	Program down load started then was either aborted or not completed because of a communication error, etc.	Attempt to re-send program.
"Prog Corrupt"	Program is corrupt or damaged.	When power is cycled, program is re-loaded from FLASH. If program stored in FLASH becomes corrupt, this error may occur.
"Flash Erased"	Flash area has been erased	When power is cycled, program is re-loaded from FLASH. If FLASH has been erased because of a system problem, this error may occur.
No error detected but program does not run.	The CR200(X) may not have adequate memory.	Check program memory requirements for data tables and variables.
No error detected, program runs but malfunctions.	Often due to a short Scan() time.	Increase the scan rate or remove lower priority instructions.

## 18.1.2 NAN and $\pm$ INF

NAN (not-a-number) and  $\pm$ INF (infinite) are data words indicating an exceptional occurrence in CR200(X) function or processing. NAN is a constant that can be used in expressions such as in [CRBASIC EXAMPLE. Using NAN in Expressions](#) (p. 153) NAN can also be used in the disable variable (DisableVar) in output processing (data storage) instructions, as indicated in [CRBASIC EXAMPLE. Using NAN in Expressions](#) p. 153.

CRBASIC EXAMPLE 21.	Using NAN in Expressions
<pre>If WindDir = NAN Then     WDFlag = True Else     WDFlag=False EndIf</pre>	

### 18.1.2.1 Measurements and NAN

A NAN indicates an invalid measurement.

#### 18.1.2.1.1 Voltage Measurements

The CR200(X) measures voltage inputs ranging from 0 to 2500 mV. Input signals that exceed this ranges result in an over range indicated by a NAN for the measured result. A voltage input not connected to a sensor is floating and the resulting measured voltage often remains near the voltage of the previous measurement. Floating measurements tend to wander in time, and could temporarily impersonate a valid measurement.

#### 18.1.2.1.2 SDI-12 Measurements

##### SDI-12 Measurements

NAN is loaded into the first SDI12Recorder () variable under these conditions:

- When busy with terminal commands.
- When the command is an invalid command.
- When the sensor aborts with CR LF and there is no data.

### 18.1.2.2 Floating Point Math, NAN, and $\pm$ INF

[TABLE. Math Expressions and CRBASIC Results](#) (p. 154) lists math expressions, their CRBASIC form, and IEEE floating point math result for the CR200(X).

Table 25. Math Expressions and CRBASIC Results		
Expression	CRBASIC Expression	Result
$0 / 0$	$0 / 0$	NAN
$\infty - \infty$	$(1 / 0) - (1 / 0)$	NAN
$(-1)^\infty$	$-1 ^ (1 / 0)$	NAN
$0 * _\infty$	$0 * (-1 * (1 / 0))$	NAN
$\pm\infty / \pm\infty$	$(1 / 0) / (1 / 0)$	NAN
$1^\infty$	$1 ^ (1 / 0)$	NAN
$0 * \infty$	$0 * (1 / 0)$	NAN
$x / 0$	$1 / 0$	NAN
$x / -0$	$1 / -0$	NAN
$-x / 0$	$-1 / 0$	NAN
$-x / -0$	$-1 / -0$	NAN
$\infty^0$	$(1 / 0) ^ 0$	1
$0^\infty$	$0 ^ (1 / 0)$	0
$0^0$	$0 ^ 0$	1

## 18.2 Communications

### 18.2.1 RS-232

Baud rate mis-match between the CR200(X) and datalogger support software is often the root of communication problems through the RS-232 port. Software settings should be set to a baud rate of 9600.

### 18.2.2 Communicating with Multiple PC Programs

A common practice is to monitor the performance of a CR200(X) using Devconfig or LoggerNet while the CR200(X) has an open connection to another copy of LoggerNet. For example, the main connection can be via RF while the performance monitoring is done via RS-232 port. This is a useful feature of the CR200(X). A problem often arises, however, in that the CR200(X) gets confused when attempting this via two different ports, from two different instances of the same PakBus® address, i.e. if LoggerNet and Devconfig have the same address.

---

**Note** LoggerNet defaults to PakBus® address 4094. Devconfig is fixed at PakBus® address 4094.

---

The solution is to change the PakBus® address in LoggerNet | Setup | Options | LoggerNet Pakbus Settings). If feasible, use PC200W or PC400 instead of Devconfig as each has a different default PakBus® address from LoggerNet.



## 18.3 Power Supply

### 18.3.1 Overview

Power supply systems may include batteries, charger/regulators, and charging sources such as solar panels or transformers. All of these components may need to be checked if the power supply is not functioning properly.

*Diagnosis and Fix Procedures* (p. 156) includes the following flowcharts:

- Battery Voltage Test
- Charging Circuit Test (when using an unregulated solar panel)
- Charging Circuit Test (when using a transformer)
- Adjusting Charging Circuit

If all of the power supply components are working properly and the system has peripheral(s) with high current drain(s) such as satellite transmitter, verify that the system's power supply provides enough power. For more information, refer to our Power Supply product literature or Application Note.

### 18.3.2 Troubleshooting at a Glance

**Symptoms:**

Possible symptoms include the CR200(X) program not executing; WatchDogCnt in the Status table displaying a non-zero number.

**Affected Equipment:**

Batteries, charger/regulators, solar panels, transformers

**Likely Cause:**

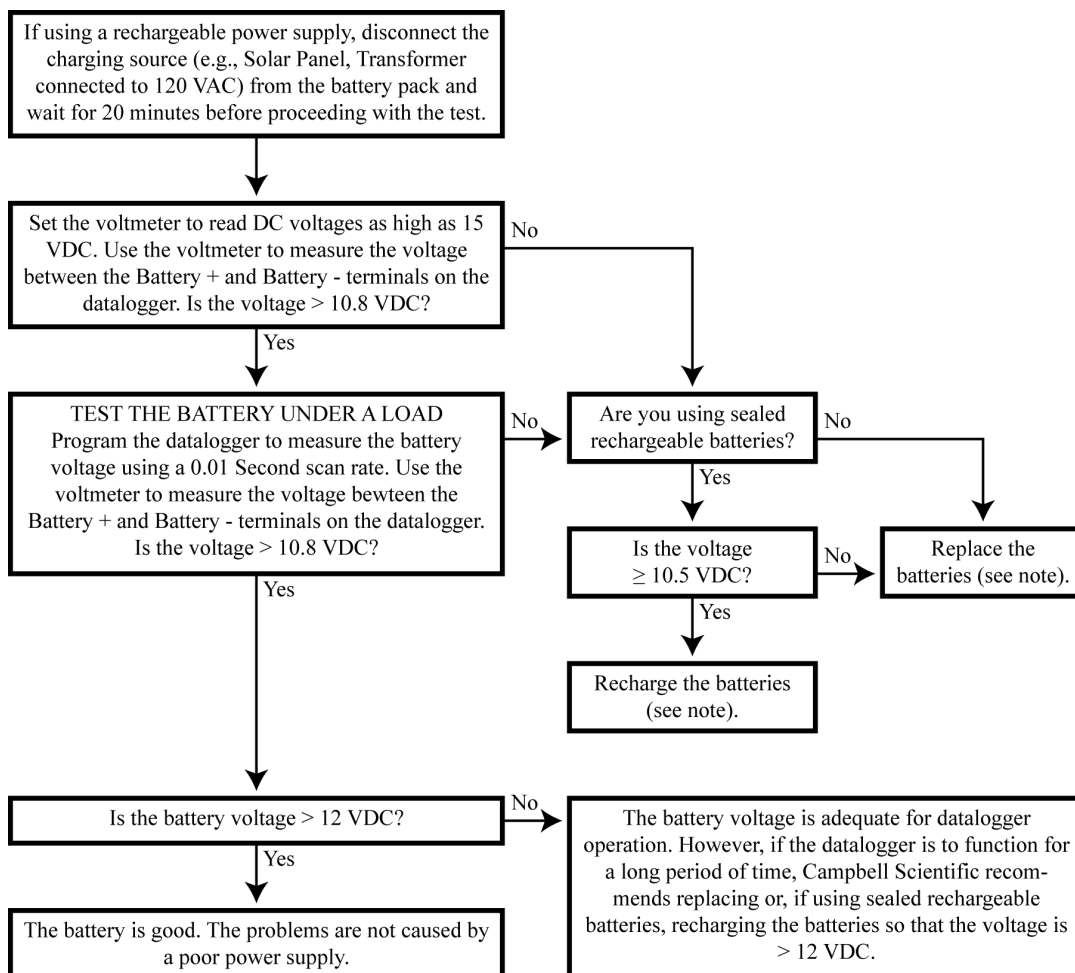
Batteries may need to be replaced or recharged; charger/regulators may need to be fixed or recalibrated; solar panels or transformers may need to be fixed or replaced.

**Required Equipment:**

Voltmeter; 5 kohm resistor and 50 ohm 1 W resistor for the charging circuit tests and to adjust the charging circuit voltage

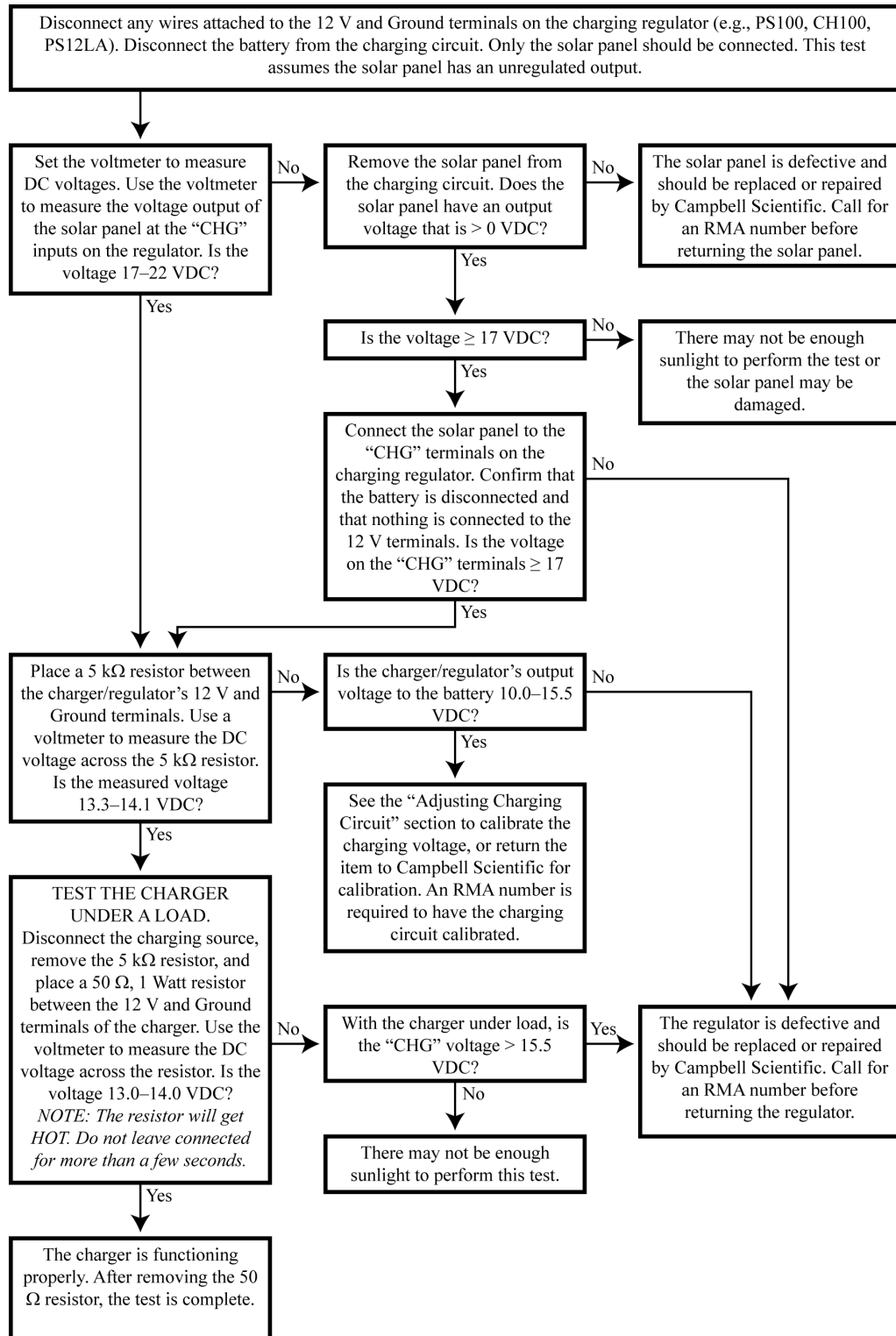
## 18.3.3 Diagnosis and Fix Procedures

### 18.3.3.1 Battery Voltage Test

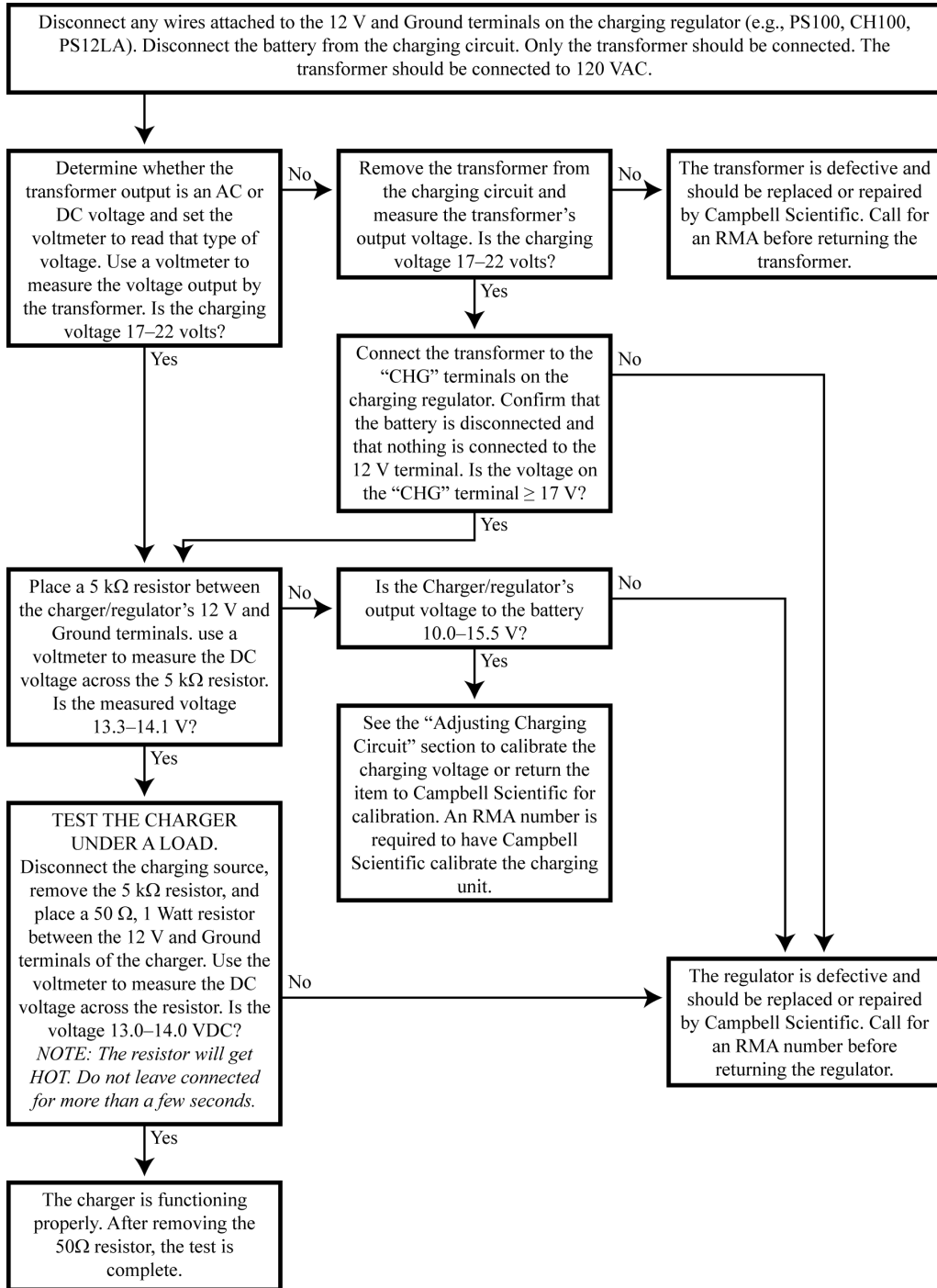


NOTE: For customers using a sealed rechargeable battery that is recharged via an unregulated solar panel or a transformer, Campbell Scientific also recommends checking the charging circuit.

### 18.3.3.2 Charging Circuit Test - Solar Panel

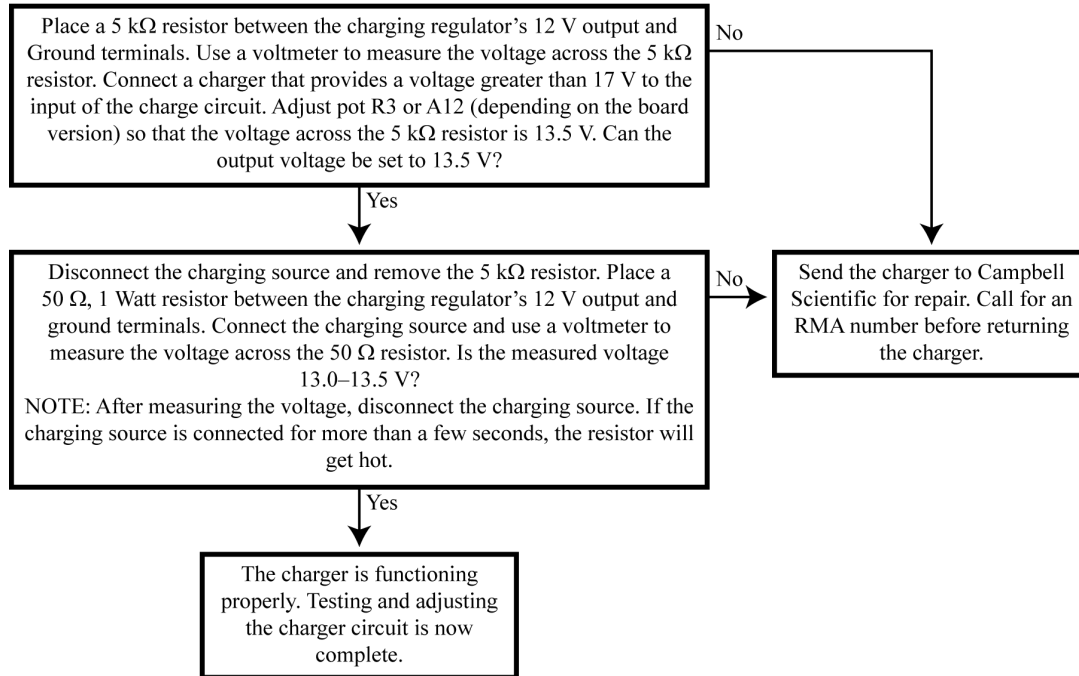


### 18.3.3.3 Charging Circuit Test - Transformer



### 18.3.3.4 Adjusting Charging Circuit Voltage

NOTE: Campbell Scientific recommends that only a qualified electronic technician perform the following procedure.





# Appendix A. Glossary

---

## A.1 Terms

AC

See *VAC* (Appendix p. 12).

A/D

Analog-to-digital conversion. The process that translates analog voltage levels to digital values.

accuracy

A measure of the correctness of a measurement. See also *Accuracy, Precision, and Resolution* (Appendix p. 13).

Amperes (Amps)

Base unit for electric current. Used to quantify the capacity of a power source or the requirements of a power consuming device.

analog

Data presented as continuously variable electrical signals.

ASCII / ANSI

Abbreviation for American Standard Code for Information Interchange / American National Standards Institute. An encoding scheme in which numbers from 0-127 (ASCII) or 0-255 (ANSI) are used to represent pre-defined alphanumeric characters. Each number is usually stored and transmitted as 8 binary digits (8 bits), resulting in 1 byte of storage per character of text.

asynchronous

The transmission of data between a transmitting and a receiving device occurs as a series of zeros and ones. For the data to be "read" correctly, the receiving device must begin reading at the proper point in the series. In asynchronous communication, this coordination is accomplished by having each character surrounded by one or more start and stop bits which designate the beginning and ending points of the information (see *Synchronous* (Appendix p. 11)).

baud rate

The speed of transmission of information across a serial interface, expressed in units of bits per second. For example, 9600 baud refers to bits being transmitted (or received) from one piece of equipment to another at a rate of 9600 bits per second. Thus, a 7 bit ASCII character plus parity bit plus 1 stop bit (total 9 bits) would be transmitted in 9/9600 sec. = .94 ms or about 1000 characters/sec. When communicating via a serial interface, the

baud rate settings of two pieces of equipment must match each other. The baud rate for CR200(X) communication should be set to 9600.

Beacon

A signal broadcasted to other devices in a PakBus® network to identify "neighbor" devices. A beacon in a PakBus® network ensures that all devices in the network are aware of other devices that are viable. If configured to do so, a clock set command may be transmitted with the beacon. This function can be used to synchronize the clocks of devices within the PakBus® network. See also [PakBus](#) (Appendix p. 7) and [Neighbor Device](#) (Appendix p. 6).

binary

Describes data represented by a series of zeros and ones. Also describes the state of a switch, either being on or off.

code

A CRBASIC program, or a portion of a program.

compile

The software process of converting a human readable program to binary machine code. CR200(X) programs are compiled by the CRBASIC Editor.

constant

A packet of CR200(X) memory given an alpha-numeric name and assigned a fixed number.

control I/O

Terminals C1 - C2 or processes utilizing these terminals.

CVI

Communications Verification Interval. The interval at which a PakBus® device verifies the accessibility of neighbors in its neighbor list. If a neighbor does not communicate for a period of time equal to 2.5 x the CVI, the device will send up to 4 Hellos. If no response is received, the neighbor is removed from the neighbor list.

CPU

Central processing unit. The brains of the CR200(X).

cr

Carriage Return



**datalogger support software**

Includes PC200W, PC400, RTDAQ, LoggerNet

**data point**

A data value which is sent to Final Storage as the result of an output processing (data storage) instruction. Strings of data points output at the same time make up a record in a data table.

**DC**

See VDC.

**DCE**

Data communications equipment. While the term has much wider meaning, in the limited context of practical use with the CR200(X), it denotes the pin configuration, gender and function of an RS-232 port. The RS-232 port on the CR200(X) and on many 3<sup>rd</sup> party telecommunications devices, such as a digital cellular modems, are DCE. Interfacing a DCE device to a DCE device requires a null-modem cable.

**desiccant**

A material that absorbs water vapor to dry the surrounding air.

**DevConfig**

Device Configuration Utility, available with LN, PC400, or from the CSI website.

**differential**

A sensor or measurement terminal wherein the analog voltage signal is carried on two leads. The phenomenon measured is proportional to the difference in voltage between the two leads.

**digital**

Numerically presented data.

**Dim**

A CRBASIC command for declaring and dimensioning variables. Variables declared with DIM remain hidden during datalogger operation.

**DTE**

While the term has much wider meaning, in the limited context of practical use with the CR200(X), it denotes the pin configuration, gender and function of an RS-232 port. The RS-232 port on the CR200(X) and on many 3<sup>rd</sup> party telecommunications devices, such as a digital cellular modems, are DCE. Attachment of a null-modem cable to a DCE device effectively converts it to a DTE device.

Earth Ground

use of a grounding rod or another suitable device to tie a system or device to the earth. Earth ground is a sink for electrical transients and possibly damaging potentials, such as those produced by a nearby lightning strike. Earth ground is the preferred reference potential for analog voltage measurements. Note that most objects have a "an electrical potential" and the potential at different places on the earth - even a few meters away - may be different.

engineering units

Units that explicitly describe phenomena, as opposed to the CR200(X) measurement units of millivolts or counts.

ESD

Electrostatic discharge

ESS

Environmental Sensor Station

excitation

Application of a precise voltage, usually to a resistive bridge circuit.

execution time

Time required to execute an instruction or group of instructions. If the execution time of a Program Table exceeds the table's Execution Interval, the Program Table is executed less frequently than programmed.

expression

A series of words, operators, or numbers that produce a value or result.

final storage

That portion of memory allocated for storing data tables with output arrays. Final storage is a ring memory, with new data overwriting the oldest data.

garbage

The refuse of the data communication world. When data are sent or received incorrectly (there are numerous reasons why this happens) a string of invalid, meaningless characters (garbage) results. Two common causes are: 1) a baud rate mismatch and 2) synchronous data being sent to an asynchronous device and vice versa.

ground

Being or related to an electrical potential of 0 Volts.

Hello Exchange

The process of verifying a node as a neighbor.

Hertz

Abbreviated Hz. Unit of frequency described as cycles or pulses per second.

IEEE4

4 byte floating point data type.

INF

Infinite or undefined. A data word indicating the result of a function is infinite or undefined.

input/output instructions

Used to initiate measurements and store the results in Input Storage or to set or read Control/Logic Ports.

integer

A number written without a fractional or decimal component. 15 and 7956 are integers. 1.5 and 79.56 are not integers.

intermediate storage

That portion of memory allocated for the storage of results of intermediate calculations necessary for operations such as averages or standard deviations. Intermediate storage is not accessible to the user.

If

line feed

loop counter

Increments by 1 with each pass through a loop.

manually initiated

Initiated by the user, as opposed to occurring under program control.

milli

The SI prefix denoting 1/1000s of a base SI unit.

Modbus

Communication protocol published by Modicon in 1979 for use in programmable logic controllers (PLCs).

modem/terminal

Any device which:

- has the ability to raise the CR200(X) ring line and put the CR200(X) in the Telecommunications Command State
- has an asynchronous serial communication port which can be configured to communicate with the CR200(X).

multi-meter

An inexpensive and readily available device useful in troubleshooting data acquisition system faults.

mV

The SI abbreviation for milliVolts.

NAN

Not a number. A data word indicating a measurement or processing error. Voltage over range, SDI-12 sensor error, and undefined mathematical results can produce NAN.

Neighbor Device

Devices in a PakBus® network that can communicate directly with an individual device without being routed through an intermediate device. See [PakBus](#) (Appendix p. 7).

NIST

National Institute of Standards and Technology

Node

Part of the description of a datalogger network when using LoggerNet. Each node represents a device that the communications server will dial through or communicate with individually. Nodes are organized as a hierarchy with all nodes accessed by the same device (parent node) entered as child nodes. A node can be both a parent and a child.

Null-modem

A device, usually a multi-conductor cable, which converts an RS-232 port from DCE to DTE or from DTE to DCE.

Ohm

The unit of resistance. Symbol is the Greek letter Omega ( $\Omega$ ). 1.0  $\Omega$  equals the ratio of 1.0 Volt divided by 1.0 Amp.

Ohms Law

Describes the relationship of current and resistance to voltage. Voltage equals the product of current and resistance ( $V = I \cdot R$ ).

on-line data transfer

Routine transfer of data to a peripheral left on-site. Transfer is controlled by the program entered in the datalogger.

output

A loosely applied term. Denotes a) the information carrier generated by an electronic sensor, b) the transfer of data from variable storage to final storage, or c) the transfer of power from the CR200(X) or a peripheral to another device.

output array

A string of data points output to Final Storage. Output occurs when the data interval and data trigger are true. The data points which complete the Array are the result of the Output Processing Instructions which are executed while the Output Flag is set.

output interval

The time interval between initiations of a particular data table record.

output processing instructions

Process data values and generate Output Arrays. Examples of Output Processing Instructions include Totalize, Maximum, Minimum, Average, etc. The data sources for these Instructions are values in Input Storage. The results of intermediate calculations are stored in Intermediate Storage. The ultimate destination of data generated by Output Processing Instructions is usually Final Storage but may be Input Storage for further processing. The transfer of processed summaries to Final Storage takes place when the Output Flag has been set by a Program Control Instruction.

PakBus

A proprietary telecommunications protocol similar in concept to internet protocol (IP). It has been developed by Campbell Scientific to facilitate communications between Campbell Scientific instrumentation.

parameter

Used in conjunction with CR200(X) program Instructions, parameters are numbers or codes which are entered to specify exactly what a given instruction is to do. Once the instruction number has been entered in a Program Table, the CR200(X) will prompt for the parameters by displaying the parameter number in the ID Field of the display.

period average

A measurement technique utilizing a high-frequency digital clock to measure time differences between signal transitions. Sensors commonly measured with period average include vibrating wire transducers and water content reflectometers.

peripheral

Any device designed for use with, and requiring, the CR200(X) (or another CSI datalogger) to operate.

precision

A measure of the repeatability of a measurement. See also [Accuracy, Precision, and Resolution](#) (Appendix p. 13).

print device

Any device capable of receiving output over pin 6 (the PE line) in a receive-only mode. Printers, "dumb" terminals, and computers in a terminal mode fall in this category.

print peripheral

See [Print Device](#) (Appendix p. 8).

processing instructions

These instructions allow the user to further process input data values and return the result to Input Storage where it can be accessed for output processing. Arithmetic and transcendental functions are included in these Instructions.

program control instructions

Used to modify the sequence of execution of Instructions contained in Program Tables; also used to set or clear flags.

Public

A CRBASIC command for declaring and dimensioning variables. Variables declared with PUBLIC can be monitored during datalogger operation.

pulse

An electrical signal characterized by a sudden increase in voltage followed by a short plateau and a sudden voltage decrease.

regulator

A device for conditioning an electrical power source. CSI regulators typically condition AC or DC voltages greater than 16 V to about 14 VDC.

resistance

A feature of an electronic circuit that impedes or redirects the flow of electrons through the circuit.

resistor

A device that provides a known quantity of resistance.

resolution

A measure of the fineness of a measurement. See also [Accuracy](#), [Precision](#), and [Resolution](#) (Appendix p. 13).

ring line (Pin 3)

Line pulled high by an external device to "awaken" the CR200(X).

Ring Memory

A memory configuration for data tables allowing the oldest data to be overwritten. This is the default setting for data tables.

RMA Number

Return Materials Authorization number. Obtain an RMA number from a Campbell Scientific applications engineer PRIOR to sending any equipment for repair.

RMS

Root mean square or quadratic mean. A measure of the magnitude of wave or other varying quantities around zero.

RS-232

Recommended Standard 232. A loose standard defining how two computing devices can communicate with each other. The implementation of RS-232 in CSI dataloggers to PC communications is quite rigid, but transparent to most users. Implementation of RS-232 in CSI datalogger to RS-232 smart sensor communications is quite flexible.

sample rate

The rate at which measurements are made. The measurement sample rate is primarily of interest when considering the effect of time skew (i.e., how close in time are a series of measurements). The maximum sample rates are the rates at which measurements are made when initiated by a single instruction with multiple repetitions.

scan (execution interval)

The time interval between initiating each execution of a given Scan interval. If the Execution Interval is evenly divisible into 24 hours (86,400 seconds), the Execution Interval is synchronized with 24 hour time, so that

the table is executed at midnight and every execution interval thereafter. The table is executed for the first time at the first occurrence of the Execution Interval after compilation. If the Execution Interval does not divide evenly into 24 hours, execution will start on the first even second after compilation.

**SDI-12**

Serial/Digital Data Interface at 1200 bps. Communication protocol for transferring data between data recorders and sensors.

**SDM**

Synchronous Device for Measurement. A processor based peripheral device or sensor that communicates with the CR200(X) via hardwire over short distance using a proprietary CSI protocol. The CR200(X) datalogger is not compatible with SDM devices.

**Send**

Denotes the program send button in LoggerNet / PC400 / RTDAQ / PC200W datalogger support software.

**serial**

A loose term denoting output or a device that outputs an electronic series of alphanumeric characters.

**SI Système Internationale**

The International System of Units.

**signature**

A number which is a function of the data and the sequence of data in memory. It is derived using an algorithm which assures a 99.998% probability that if either the data or its sequence changes, the signature changes.

**single-ended**

Denotes a sensor or measurement terminal where in the analog voltage signal is carried on a single lead, which is measured with respect to ground.

**skipped scans**

Occurs when the CR200(X) program is too long for the scan interval. Skipped scans can cause errors in pulse measurements.

**SP**

Space



state

Whether a device is on or off.

string

A datum consisting of alpha-numeric characters. CR200(X) dataloggers do not support the STRING variable type.

support software

Includes PC200W, PC400, RTDAQ, LoggerNet.

synchronous

The transmission of data between a transmitting and receiving device occurs as a series of zeros and ones. For the data to be "read" correctly, the receiving device must begin reading at the proper point in the series. In synchronous communication, this coordination is accomplished by synchronizing the transmitting and receiving devices to a common clock signal (see [Asynchronous](#) (Appendix p. 1)).

throughput

The throughput rate is the rate at which a measurement can be made, scaled to engineering units, and the reading stored in a data table. The CR200(X) has the ability to scan sensors at a rate exceeding the throughput rate. The primary factor affecting throughput rate is the amount of processing specified by the user. In a CR200(X) all processing called for by an instruction must be completed before moving on the next instruction.

TLL

Transistor - Transistor Logic. A serial protocol using 0V and 5V as logic signal levels.

toggle

To reverse the current power state.

UPS

Uninterruptible power supply. A UPS can be constructed for most datalogger applications using AC line power, an AC/AC or AC/DC wall adapter, a charge controller, and a rechargeable battery.

User Program

The CRBASIC program written by the CR200(X) user in CRBASIC Editor or Short Cut.

variable

A packet of CR200(X) memory given an alpha-numeric name, which holds a potentially changing number or string.

VAC

Volts Alternating Current. Mains or grid power is high-level VAC, usually 110 VAC or 220 VAC at a fixed frequency of 50 Hz or 60 Hz. High-level VAC is used as a primary power source for Campbell Scientific power supplies. Do not connect high-level VAC directly to the CR200(X). The CR200(X) measures varying frequencies of low-level VAC in the range of  $\pm 20$  VAC.

VDC

Volts Direct Current. The CR200(X) operates with a nominal 12 VDC power supply. It can supply nominal 12 VDC and variable excitation in the 5 VDC and 2.5 VDC range. It measures analog voltage in the 0 – 2.5 VDC range and pulse voltage in the  $\pm 20$  VDC range.

volt meter

An inexpensive and readily available device useful in troubleshooting data acquisition system faults.

Volts

SI unit for electrical potential.

watch dog timer

An error checking system that examines the processor state, software timers, and program related counters when the datalogger is running its program. If the processor has bombed or is neglecting standard system updates or if the counters are outside the limits, the watchdog timer resets the processor and program execution. Voltage surges and transients can cause the watchdog timer to reset the processor and program execution. When the watchdog timer resets the processor and program execution, an error count is incremented in the watchdogtimer entry of the status table. A low number (1 to 10) of watchdog timer resets is of concern, but normally indicates the user should just monitor the situation. A large number ( $>10$ ) of error accumulating over a short period of time should cause increasing alarm since it indicates a hardware or software problem may exist. When large numbers of watchdog timer resets occur, consult with a Campbell Scientific applications engineer. See [WatchdogErrors](#) p. 151 for more information.

weather tight

Describes an instrumentation enclosure impenetrable by common environmental conditions. During extraordinary weather events, however, seals on the enclosure may be breached.

XML

Extensible Markup Language.

## A.2 Concepts

### A.2.1 Accuracy, Precision, and Resolution

Three terms often confused are accuracy, precision, and resolution. **Accuracy** is a measure of the correctness of a single measurement, or the group of measurements in the aggregate. **Precision** is a measure of the repeatability of a group of measurements. **Resolution** is a measure of the fineness of a measurement. Together, the three define how well a data acquisition system performs. To understand how the three relate to each other, consider "target practice" as an analogy. [FIGURE. Accuracy, Precision, and Resolution](#) (p. 13) shows four targets. The bull's eye on each target represents the absolute correct measurement. Each shot represents an attempt to make the measurement. The diameter of the projectile represents resolution.

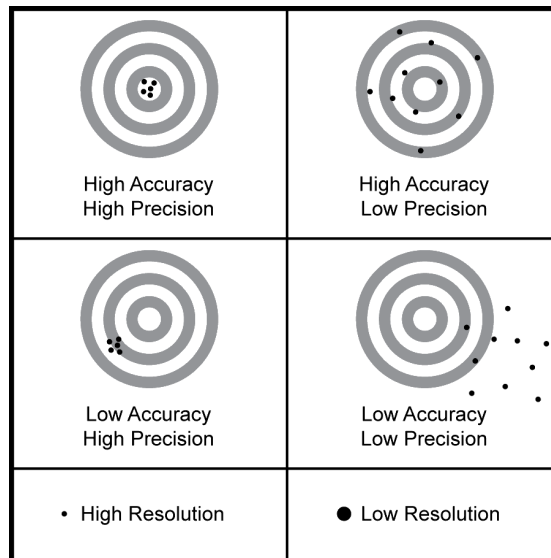


Figure 53: Accuracy, Precision, and Resolution

The objective of a data acquisition system should be high accuracy, high precision, and to produce data with resolution as high as appropriate for a given application.



## Appendix B. Status Table and Settings

The CR200(X) status table contains system operating status information accessible via PC software DevConfig, LoggerNet, PC400, RTDAQ, or PC200W. [TABLE. Common Uses of the Status Table](#) (Appendix p. 15) lists some of the more common uses of status table information. [TABLE. Status Table Fields and Descriptions](#) (Appendix p. 16) is a comprehensive list of status table variables with brief descriptions.

Status Table information is easily viewed by going to LoggerNet / PC400 / RTDAQ / PC200W | Datalogger | Station Status. However, be aware that information presented in Station Status is not automatically updated. Click the refresh button each time an update is desired. Alternatively, use the Numeric displays of the connect screen to show critical values and have these update automatically, or use Devconfig, which polls the status table at regular intervals without use of a refresh button. Note that a lot of comms and other activity is needed to generate the Status Table, so if the CR200(X) is very tight on time, just getting the Status Table itself repeatedly could push timing over the edge and cause skipped scans.

Through the continued development of the operating system, the status table has become quite large. A separate settings table has been introduced to slow the growth of the status table. To maintain backward compatibility, settings first included in the status table have been retained, but are also included in the settings editor.

<i>Feature or Suspect Constituent</i>	<i>Status Field(s) to Consult</i>
Reset Data Tables	ResetTables (Enter 8888)
Operating System	OSVersion
	OSDate
	OSSignature
	WatchdogCnt
CRBASIC Program	ProgName
	ProgSig
	VarOutOfBounds
	SkipScan
EEPROM Error	TrapCode
Power Supply	Battery
	WatchdogCnt
PakBus	PakBusAddress
Radio	RfInstalled
	RfNeAddress
	RfHopSeq
	RfPwrMode
	Rf_ForceOn
	Rf_Protocol
	RfSignalLevel

<b>Table 26. Status Table Fields and Descriptions</b>						
<i>Status Table Fieldname</i>	<i>Description</i>	<i>Variable Type</i>	<i>Default</i>	<i>Normal Range</i>	<i>User can change?</i>	<i>Info Type</i>
RecNum	Increments for successive status table data records		–	0 to 2 <sup>32</sup> –	–	
TimeStamp	Time the record was generated	Time	–	–	–	
ProgName	Name of current (running) program.	String	–	–	–	Status
OSVersion	Version of the Operating System	String	–	–	–	Status
OSDate	Date OS was released.	String	–	–	–	Status
ProgName	Name of current (running) program.	String	–	–	–	Status
ProgSig	Signature of the current running program file including comments. Does not change with operating system changes.	Integer	–	–	–	Status
CalOffset	Calibration table of single ended offset values. Each integration / range combination has a single ended offset associated with it. These numbers are updated by the background slow sequence if needed in the program.	Integer array of 18	–	close to 0	–	Calib
PakBusAddress <sup>1</sup>	CR200(X) PakBus address.	String	1	1 to 3999	Yes	Config PB
RfInstalled	Indicates the model of RF radio installed	Integer	–	–	–	Status
RfNetAddr	Address of the RF network	Integer	0	0 to 63	Yes	ConfigPB
RfAddress	Address of the RF radio	Integer	0	0 to 1023	Yes	ConfigPB
RfHopSeq	RF hopping sequence for the RF network	Integer	0	0 to 6	Yes	ConfigPB
RfPwrMode	Powermode of the RF radio	String	–	–	–	ConfigPB
Rf_ForceOn	Any non-zero number will override the current power mode and set it on.	Integer	0		Yes	ConfigPB
Rf_Protocol	Protocol to use in the RF network	Integer	1	1 or 2	Yes	ConfigPB
RfSignalLevel	RF signal level		–		–	Status

Table 26. Status Table Fields and Descriptions						
RfRXPakBusCnt						Status
VarOutOfBounds	Number of times an array was accessed out of bounds <sup>2</sup>	Integer	0	0	Can Reset = 0	Error
SkipScan	Number of skipped scans that have occurred because the datalogger was not finished with the previous scan	Integer	0	–	Can Reset = 0	Error
TrapCode	Indicates a problem with datalogger memory	Integer	0	0 or 16	–	Error
WatchDogCnt	Number of Watchdog errors that have occurred while running this program <sup>3</sup>	Integer	0	0	Can Reset = 0	Error
ResetTables	Resets all the data tables in the datalogger when changed to 8888	Integer	0	_0	Can Reset = 8888	Status
BattVoltage	Current value of the battery voltage. Measurement is made in the background calibration.	Float	–	7-16 Volts	–	Measure
ProgSignature	Signature of the current running program file including comments. Does not change with operating system changes.	Integer	–	–	–	Status

1. Pak Bus Addresses 1 to 4094 are valid. Addresses  $\geq 4000$  are generally used for a PC by PC200W, RTDAQ, PC400, or LoggerNet.
2. Watchdog errors are automatically reset upon compiling a new program.
3. The Variable Out-of-Bounds error occurs when a program tries to write to an array variable outside of its declared size. A programming error causes this, so it should not be ignored. When the datalogger detects that a write outside of an array is being attempted it does not perform the write and increments the VOOB in the status table. The compiler and pre-compiler can only catch things like reps too large for an array etc. If an array is used in a loop or expression the pre-compiler does not (in most cases cannot) check to see if an array is accessed out of bounds (i.e. accessing an array with a variable index such as `arr(index) = arr(index-1)`, where index is a variable).

Table 27. CR200(X) Settings																				
Settings are accessed through Campbell Scientific's Device Configuration Utility (DevConfig) for direct serial connection, or through PakBusGraph for most telecommunications options.																				
Setting	Description	Default Entry																		
Max Packet Size	Specified the maximum size packet in bytes that should ever be sent to the device.	1000																		
PakBus Address	<p>This setting specifies the PakBus® address for this device. The value for this setting must be chosen such that the address of the device is unique in the scope of the datalogger network. Duplication of PakBus® addresses in two or more devices can lead to failures and unpredictable behavior in the PakBus® network. The following values are the default addresses of various types of software and devices and should probably be avoided:</p> <table><tr><td><b>Device</b></td><td><b>PB Address</b></td></tr><tr><td>LoggerNet</td><td>4094</td></tr><tr><td>PC400</td><td>4093</td></tr><tr><td>PC200</td><td>4092</td></tr><tr><td>Visual Weather</td><td>4091</td></tr><tr><td>RTDAQ</td><td>4090</td></tr><tr><td>DevConfig</td><td>4089</td></tr><tr><td>NL100</td><td>678</td></tr><tr><td>All other devices</td><td>1</td></tr></table>	<b>Device</b>	<b>PB Address</b>	LoggerNet	4094	PC400	4093	PC200	4092	Visual Weather	4091	RTDAQ	4090	DevConfig	4089	NL100	678	All other devices	1	1
<b>Device</b>	<b>PB Address</b>																			
LoggerNet	4094																			
PC400	4093																			
PC200	4092																			
Visual Weather	4091																			
RTDAQ	4090																			
DevConfig	4089																			
NL100	678																			
All other devices	1																			
Radio Installed	Specifies the model number of the MaxStream radio if it is recognized by the datalogger. It will have a value of zero if there is no radio recognized by the datalogger.																			
RF Network Address	Specifies the radio network address of the built-in radio. This setting should be set to match the network address for the RF400 base used to communicate with the datalogger.	0																		
RF Address	Specifies the address of the built-in radio. This value should generally be set to match the same as the address of the RF400 base used to communicate with the datalogger unless that base is used to communicate with non-PakBus® dataloggers (such as a CR10X).	0																		
RF Hop Sequence	Specifies the hopping sequence that will be used for the built-in radio. This value should be set to match the value of the same setting for the RF400 base station used to communicate with this datalogger	0																		
Ignore RF Power Mode	If set to true (non-zero value), the RF power mode setting will be ignored and the radio will be powered on all of the time.	0																		
RF Power Mode	<p>This setting specifies the power wait state that will be set in the built-in radio. This setting must be set so that the standby mode of the radio is consistent with any other CR200(X) or RF400/RF410/Rf415 with which this station must communicate.</p> <p>The power modes supported include the following:</p> <p>No Radio</p> <p>Indicates that there is no radio hardware recognized by the datalogger. The datalogger will not accept this value as an input, but will report it if there is no radio hardware detected.</p> <p>Always On</p> <p>Indicates that the radio receiver circuit should always be ready to receive data.</p>	50																		



**Table 27. CR200(X) Settings**

Settings are accessed through Campbell Scientific's Device Configuration Utility (DevConfig) for direct serial connection, or through PakBusGraph for most telecommunications options.

<b>Setting</b>	<b>Description</b>	<b>Default Entry</b>
	<p>1 Sec Indicates that the radio receiver is to use the one second duty cycle.</p> <p>8 Sec Indicates that the radio receiver is to use the eight second duty cycle.</p> <p>1 Sec with Long Header Indicates that the radio receiver will use the one second duty cycle and that the radio will transmit a one second long header before sending data if the time since last transmission is over one second.</p> <p>8 Sec with Long Header Indicates that the radio receiver will use the eight second duty cycle and that the radio will transmit an eight second long header if the time since the last transmission is greater than eight seconds.</p> <p>Pin Enabled Indicates that the radio receiver is to be turned on only when the RF_ForceOn status table variable is enabled.</p>	
Company	Identifies the manufacturer	CSI
PakCtrl Command Codes	Identifies the list of PakBus PakCtrl Interface command codes that the logger will accept.	2 7 8 9 12
BMP5 Command Codes	Identifies the list of PakBus BMP5 Interface command codes that the datalogger will accept.	9 23 24 26 27 28 29 30
Model Number	Identifies the model number assigned to this datalogger	CR2xx
RF Protocol	<p>Identifies the radio protocol that will be used for the CR2xx. In order to be compatible with other CR2xx and RF400 type devices, the default value of transparent must be used. The following values are supported:</p> <p>Transparent This mode is compatible with older CR205 and RF400 operating systems.</p> <p>PakBus Aware This mode can be used in networks involving RF401/RF411/RF416 hardware or other, newer CR20x, CR211/CR216 devices and makes use of the retry capability inherent in the MaxStream radios. This mode is not compatible with the older radios.</p>	Transparent



# Appendix C. Serial Port Pin Outs

## C.1 RS-232 Communications Port

### C.1.1 Pin-Out

Pin configuration for the CR200(X) RS-232 9-pin port is listed in [TABLE. CR200\(X\) RS-232 Pin-Out](#) p. 21.

The CR200(X) RS-232 port is a DCE (Data Communication Equipment) device. A limited version of the RS-232 port is supported with no hardware flow control. The most common use of the Datalogger's RS-232 port is a connection to a computer DTE device. A standard DB9-to-DB9 cable can connect the computer DTE device to the Datalogger DCE device.

The CR200(X) RS-232 port is not electrically isolated. Connection to an AC powered computer may cause ground loops leading to measurement problems.

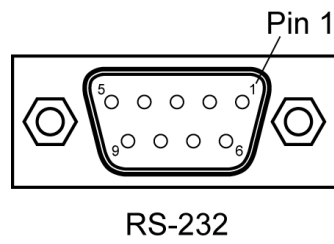
Maximum input =  $\pm 25$  V

Maximum Output =  $\pm 13$  V

Typical Output =  $\pm 5.4$  V

The following table describes the Datalogger's RS-232 pin function with standard DCE naming notation.

Table 28. CR200(X) RS-232 Pin-Out				
<b>PIN:</b> Pin number				
<b>O:</b> Signal Out of the CR200(X) to a RS-232 device				
<b>I:</b> Signal Into the CR200(X) from a RS-232 device				
<b>PIN</b>	<b>DCE Function</b>	<b>Logger Function</b>	<b>I/O</b>	<b>Description</b>
1		N/A		Not Connected
2	TX	TXD	O	Asynchronous data Transmit
3	RX	RXD	I	Asynchronous data Receive
4		N/A		Not Connected
5	GND	GND		Ground
6	DSR	DTR	O	Data Terminal Ready
7		N/A		Not Connected
8	CTS	RTS	O	Request to send
9		N/A		Not Connected



# Appendix D. ASCII / ANSI Table

American Standard Code for Information Interchange (ASCII) / American National Standards Institute (ANSI)

Decimal and Hexadecimal Codes and Characters Used with CR200(X) Tools

Dec	Hex	Keyboard Display Char	LoggerNet Char	Hyper-Terminal Char	Dec	Hex	Keyboard Display Char	LoggerNet Char	Hyper-Terminal Char
0	0		NULL	NULL	128	80		€	Ç
1	1		□	☺	129	81		□	ü
2	2		□	☹	130	82		,	é
3	3		□	♥	131	83		f	â
4	4		□	♦	132	84		„	ä
5	5		□	♣	133	85		...	à
6	6		□	♠	134	86		†	â
7	7		□	•	135	87		‡	ç
8	8		□	■	136	88		^	ê
9	9			ht	137	89		%	ë
10	a		lf	lf	138	8a		Š	è
11	b		□	vt	139	8b		ˆ	ï
12	c		□	ff	140	8c		Œ	î
13	d		cr	cr	141	8d		□	ì
14	e		□	♪	142	8e		Ž	Ä
15	f		□	☼	143	8f		□	Å
16	10		□	►	144	90		□	É
17	11		□	◄	145	91		'	æ
18	12		□	↕	146	92		'	Æ
19	13		□	!!	147	93		"	ô
20	14		□	¶	148	94		"	ö
21	15		□	§	149	95		•	ò
22	16		□	—	150	96		-	û
23	17		□	↕	151	97		-	ù
24	18		□	↑	152	98		~	ÿ
25	19		□	↓	153	99		™	Ö
26	1a		□	→	154	9a		§	Ü
27	1b		□		155	9b		>	ç

Dec	Hex	Keyboard Display Char	LoggerNet Char	Hyper-Terminal Char	Dec	Hex	Keyboard Display Char	LoggerNet Char	Hyper-Terminal Char
28	1c		□	└	156	9c		œ	£
29	1d		□	↔	157	9d		□	¥
30	1e		□	▲	158	9e		ž	Pt
31	1f		□	▼	159	9f		Ÿ	f
32	20		SP	SP	160	a0			á
33	21	!	!	!	161	a1		ı	í
34	22	"	"	"	162	a2		¢	ó
35	23	#	#	#	163	a3		£	ú
36	24	\$	\$	\$	164	a4		¤	ñ
37	25	%	%	%	165	a5		¥	Ñ
38	26	&	&	&	166	a6		ı	ª
39	27	'	'	'	167	a7		§	º
40	28	(	(	(	168	a8		ˆ	ı
41	29	)	)	)	169	a9		©	ı
42	2a	*	*	*	170	aa		ª	ı
43	2b	+	+	+	171	ab		«	½
44	2c	,	,	,	172	ac		ı	¼
45	2d	-	-	-	173	ad			ı
46	2e	.	.	.	174	ae		®	«
47	2f	/	/	/	175	af		—	»
48	30	0	0	0	176	b0		°	⋮
49	31	1	1	1	177	b1		±	⋮
50	32	2	2	2	178	b2		²	■
51	33	3	3	3	179	b3		³	
52	34	4	4	4	180	b4		´	ı
53	35	5	5	5	181	b5		μ	ı
54	36	6	6	6	182	b6		¶	ı
55	37	7	7	7	183	b7		·	ı
56	38	8	8	8	184	b8		˘	ı
57	39	9	9	9	185	b9		¹	ı
58	3a	:	:	:	186	ba		º	ı
59	3b	;	;	;	187	bb		»	ı
60	3c	<	<	<	188	bc		¼	ı
61	3d	=	=	=	189	bd		½	ı
62	3e	>	>	>	190	be		¾	ı

Dec	Hex	Keyboard Display Char	LoggerNet Char	Hyper-Terminal Char	Dec	Hex	Keyboard Display Char	LoggerNet Char	Hyper-Terminal Char
63	3f	?	?	?	191	bf		ı	ƿ
64	40	@	@	@	192	c0		À	Ł
65	41	A	A	A	193	c1		Á	ł
66	42	B	B	B	194	c2		Â	Ṭ
67	43	C	C	C	195	c3		Ã	Ṫ
68	44	D	D	D	196	c4		Ä	—
69	45	E	E	E	197	c5		Å	†
70	46	F	F	F	198	c6		Æ	ƒ
71	47	G	G	G	199	c7		Ç	‡
72	48	H	H	H	200	c8		È	℔
73	49	I	I	I	201	c9		É	℞
74	4a	J	J	J	202	ca		Ê	℥
75	4b	K	K	K	203	cb		Ë	℥
76	4c	L	L	L	204	cc		Ì	‡
77	4d	M	M	M	205	cd		Í	=
78	4e	N	N	N	206	ce		Î	‡
79	4f	O	O	O	207	cf		Ï	±
80	50	P	P	P	208	d0		Ð	℥
81	51	Q	Q	Q	209	d1		Ñ	〒
82	52	R	R	R	210	d2		Ò	π
83	53	S	S	S	211	d3		Ó	℔
84	54	T	T	T	212	d4		Ô	℔
85	55	U	U	U	213	d5		Õ	ƒ
86	56	V	V	V	214	d6		Ö	π
87	57	W	W	W	215	d7		×	‡
88	58	X	X	X	216	d8		Ø	≠
89	59	Y	Y	Y	217	d9		Ù	Ƶ
90	5a	Z	Z	Z	218	da		Ú	ƴ
91	5b	[	[	[	219	db		Û	■
92	5c	\	\	\	220	dc		Ü	■
93	5d	]	]	]	221	dd		Ý	■
94	5e	^	^	^	222	de		Þ	■
95	5f	_	_	_	223	df		ß	■
96	60	`	`	`	224	e0		à	α
97	61	a	a	a	225	e1		á	β

Dec	Hex	Keyboard Display Char	LoggerNet Char	Hyper- Terminal Char	Dec	Hex	Keyboard Display Char	LoggerNet Char	Hyper- Terminal Char
98	62	b	b	b	226	e2		â	Г
99	63	c	c	c	227	e3		ã	π
100	64	d	d	d	228	e4		ä	Σ
101	65	e	e	e	229	e5		å	σ
102	66	f	f	f	230	e6		æ	μ
103	67	g	g	g	231	e7		ç	τ
104	68	h	h	h	232	e8		è	Φ
105	69	i	i	i	233	e9		é	Θ
106	6a	j	j	j	234	ea		ê	Ω
107	6b	k	k	k	235	eb		ë	δ
108	6c	l	l	l	236	ec		ì	∞
109	6d	m	m	m	237	ed		í	φ
110	6e	n	n	n	238	ee		î	ε
111	6f	o	o	o	239	ef		ï	∩
112	70	p	p	p	240	f0		ð	≡
113	71	q	q	q	241	f1		ñ	±
114	72	r	r	r	242	f2		ò	≥
115	73	s	s	s	243	f3		ó	≤
116	74	t	t	t	244	f4		ô	∫
117	75	u	u	u	245	f5		ö	∫
118	76	v	v	v	246	f6		ø	÷
119	77	w	w	w	247	f7		÷	≈
120	78	x	x	x	248	f8		ø	°
121	79	y	y	y	249	f9		ù	·
122	7a	z	z	z	250	fa		ú	·
123	7b	{	{	{	251	fb		û	√
124	7c				252	fc		ü	ⁿ
125	7d	}	}	}	253	fd		ý	²
126	7e	~	~	~	254	fe		þ	■
127	7f		□	△	255	ff		ÿ	



# ***Appendix E. Antenna Usage and Compliance***

---

## **E.1 Use of Antenna with CR200(X)**

An FCC authorized antenna is required for use with CR200(X) models that have a built-in radio. Several models are available from Campbell Scientific.

These antennas have been tested at an authorized FCC open-field test site and are certified to be in compliance with FCC emissions limits. All antennas or antenna cables have an SMA female connector for connection to the CR200(X). The use of an unauthorized antenna could cause transmitted field strengths in excess of FCC rules, interfere with licensed services, and result in FCC sanctions against user.

## **E.2 Part 15 FCC Compliance Warning**

Changes or modifications to the CR200(X) series radio systems not expressly approved by Campbell Scientific, Inc. could void the user's authority to operate this product.

Note: This equipment has been tested and found to comply with the limits for a Class B digital device, pursuant to part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference in a residential installation. This equipment generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instructions, may cause harmful interference to radio communications. However, there is no guarantee that interference will not occur in a particular installation. If this equipment does cause harmful interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one or more of the following measures:

- Reorient or relocate the receiving antenna.
- Increase the separation between the equipment and receiver.
- Connect the equipment into an outlet on a circuit different from that to which the receiver is connected.
- Consult the dealer or an experienced radio/TV technician for help.

This device complies with part 15 of the FCC Rules. Operation is subject to the following two conditions:

1. This device may not cause harmful interference, and
2. This device must accept any interference received, including interference that may cause undesired operation.

## E.2.1 Use of Approved Antennas

### FCC OET Bulletin No. 63 (October 1993)

Changing the antenna on a transmitter can significantly increase, or decrease, the strength of the signal that is ultimately transmitted. Except for cable locating equipment, the standards in Part 15 are not based solely on output power but also take into account the antenna characteristics. Thus, a low power transmitter that complies with the technical standards in Part 15 with a particular antenna attached can exceed the Part 15 standards if a different antenna is attached. Should this happen it could pose a serious interference problem to authorized radio communications such as emergency, broadcast, and air-traffic control communications.

***Caution: In order to comply with the FCC RF exposure requirements, the CR200(X) series may be used only with approved antennas that have been tested with the on-board radio and a minimum separation distance of 20 cm must be maintained from the antenna to any nearby persons.***

# Index

---

## A

Abbreviations • 90  
ac • 1  
ac Excitation • 6  
ac Sine Wave • 7  
Accuracy • 3, 1, 13  
Address • 16  
Address -- Modbus • 141  
Address -- SDI-12 • 112  
Amps (Amps) • 1  
Analog • 6, 25, 1  
Analog Input Range • 35, 40  
Analog Measurement • 153  
Analog Sensor • 46  
Analog Sensors • 6, 35  
AND Operator • 88  
ANSI • 1, 23  
Arithmetic • 87  
Arithmetic Functions • 102  
Array • 75, 86, 88, 7  
Asynchronous Communication • 8, 1

## B

Background Calibration • 41  
Backup Battery • 33  
Battery Backup • 33  
Baud • 11, 59, 105, 154  
Baud Rate • 105, 108, 1  
Beacon • 135, 2  
Beginner Software • 11  
Binary • 2  
Binary Control • 49  
Binary Format • 71  
Boolean • 75  
Bridge • 6, 7, 41  
Bridge Measurement • 7  
Budget • 53, 118, 119

## C

Cable Length • 46  
Calibration • 33  
Care • 32, 147  
Ccontrol I/O • 37, 2  
CE Compliance • 37  
Charging Circuit • 158, 159, 1  
Circuit • 51  
Clients • 145  
Clock Accuracy • 37  
Clock Function • 104  
Clock Synchronization • 11

Code • 2  
Coil • 140  
Collecting Data • 18  
COM Port Connection • 11  
Commands - SDI-12 • 118  
Comment • 69  
Communication • 11, 18, 31, 131, 154  
Communications Ports • 28  
Compile Results • 16  
Configuration • 59  
Connection • 4, 11, 25  
Conserving Code Space • 88  
Constant • 73, 76, 2  
Constant -- Predefined • 76  
Constant Declaration • 93  
Control • 50, 96  
Control Instructions • 8  
Control Output Expansion • 49  
Control Peripheral • 49  
Control Port • 8  
cr • 2  
CR10X • 105  
CR23X • 105  
CR510 • 105  
CRBASIC Editor • 70  
CRBASIC Program • 11, 18  
CRBASIC Programming • 69

## D

Data Acquisition System -- Components • 3  
Data Acquisition System -- Data Retrieval • 4  
Data Acquisition System -- Datalogger • 4  
Data Acquisition System -- Sensor • 3, 37  
Data Collection • 4, 18  
Data Format • 132  
Data Monitoring • 11, 18  
Data Point • 3  
Data Retrieval • 131, 132  
Data Storage • 30, 81, 94  
Data Storage -- Trigger • 126  
Data Table • 11, 77, 80, 90, 94  
Data Table Access • 106  
Data Table Management • 106  
Data Table Modifier • 94  
Data Table Name • 73  
Data Type • 75  
Data Types, NAN, and  $\pm$ INF • 153  
Datalogger • 4  
Datalogger Support Software • 34, 3  
DC • 3  
DC Excitation • 37  
DCE • 28, 3, 6

Debugging • 151  
Declaration • 73, 77, 93  
Declaration -- Data Table • 94  
Declaration -- Modbus • 140  
Desiccant • 32, 3  
DevConfig • 59, 3  
Device Configuration • 59  
Device Map • 138  
Diagnosis -- Power Supply • 156  
Diagnostics • 98  
Differential • 3  
Digital • 3  
Digital I/O • 8, 25, 37, 45, 49, 99  
Dimension • 75  
Disable Variable • 81, 82, 125, 153  
DisableVar • 125, 153  
Documentation • 69  
DTE • 28, 3, 6  
Durable Settings • 67

## E

Earth Ground • 27, 55, 4  
Editor • 13  
Editor -- Short Cut • 70  
Enclosures • 147  
Engineering Units • 4  
Error • 153  
Error -- Analog Measurement • 38, 57  
Errors • 151  
ESD • 27, 4, 12  
ESD Protection • 55, 56  
Evapotranspiration • 95  
Example • 10  
Example Program • 10, 13  
Excitation • 4  
Execution • 83  
Execution Interval • 83  
Execution Time • 4  
Expression • 86, 87, 4  
Expression -- Logical • 88  
External Power Supply • 28

## F

Final Storage • 4  
Firmware • 29  
Flag • 75, 76, 141  
Floating Point Arithmetic • 87  
Format -- Numerical • 71  
Forward • 1  
Frequency • 42  
Function Codes -- Modbus • 141

## G

Gain • 38, 86  
Garbage • 4  
Gas-Discharge Tubes • 55  
Generator • 13, 70  
Glossary • 1  
GOES • 108  
Ground • 27, 33, 55, 56, 4

## H

Half Bridge • 41  
Half Duplex • 5  
Hello Exchange • 5  
Hello Request • 135  
Hello-Message • 135  
Hertz • 5  
Hexadecimal • 71  
High-Frequency • 44  
Humidity • 32

## I

I/O Port • 49  
ID • 62  
IEEE4 • 75, 5  
Infinite • 153  
Initiate Telecommunications • 109, 132  
Input Channel • 6  
Input Range • 37, 40  
Input/Output Instructions • 5  
Installation • 4  
InstrucIntions • 96  
Instruction • 84  
Instruction Times • 98  
Instructions -- ABS • 102  
Instructions -- ACOS • 101  
Instructions -- Alias • 72, 73, 77, 85, 93  
Instructions -- AND • 100  
Instructions -- ASIN • 101  
Instructions -- ATN • 101  
Instructions -- ATN2 • 101  
Instructions -- Average • 94  
Instructions -- Avgspa • 103  
Instructions -- Battery • 155, 156  
Instructions -- BeginProg ... EndProg • 96  
Instructions -- Call • 96  
Instructions -- CallTable • 96  
Instructions -- ClockSet • 104  
Instructions -- Const • 93  
Instructions -- COS • 101  
Instructions -- CovSpa • 103  
Instructions -- DataInterval • 81, 94  
Instructions -- DataTable ... EndTable • 94  
Instructions -- Delay • 96

Instructions -- Dim • 93, 3  
 Instructions -- Do ... Loop • 96  
 Instructions -- EXP • 102  
 Instructions -- FieldNames • 94  
 Instructions -- FIX • 102  
 Instructions -- For ... Next • 96  
 Instructions -- FRAC • 102  
 Instructions -- GetValue • 105  
 Instructions -- GOESData • 108  
 Instructions -- GOESGPS • 108  
 Instructions -- GOESSetup • 108  
 Instructions -- GOESStatus • 108  
 Instructions -- IfTime • 104  
 Instructions -- IIF • 100  
 Instructions -- INT • 102  
 Instructions -- LOG • 102  
 Instructions -- LOG10 • 102  
 Instructions -- Maximum • 94  
 Instructions -- MaxSpa • 103  
 Instructions -- Minimum • 94  
 Instructions -- MinSpa • 103  
 Instructions -- MOD • 102  
 Instructions -- ModBusMaster • 107  
 Instructions -- ModBusSlave • 107  
 Instructions -- NOT • 100  
 Instructions -- OR • 100  
 Instructions -- PeriodAvg • 99  
 Instructions -- PortGet • 99  
 Instructions -- PortSet • 99  
 Instructions -- Public • 93, 8  
 Instructions -- PulseCount • 98  
 Instructions -- Randomize • 104  
 Instructions -- ReadSendGetInfo • 105  
 Instructions -- RealTime • 104  
 Instructions -- RectPolar • 102  
 Instructions -- RMSSpa • 103  
 Instructions -- RND • 104  
 Instructions -- Sample • 94  
 Instructions -- Scan ... NextScan • 96  
 Instructions -- Select Case ... Case ... Case Is ... Case  
 Else ... EndSelect • 96  
 Instructions -- SendData • 105  
 Instructions -- SerialInput • 105  
 Instructions -- SetStatus • 106  
 Instructions -- SetValue • 105  
 Instructions -- SGN • 102  
 Instructions -- SIN • 101  
 Instructions -- StdDev • 94  
 Instructions -- StdDevSpa • 103  
 Instructions -- TAN • 101  
 Instructions -- Totalize • 94  
 Instructions -- Units • 93  
 Instructions -- While ... Wend • 96  
 Instructions -- WindVector • 95  
 Instructions -- XOR • 100  
 Instructions Sqr • 102

Integer • 5  
 Integration • 40  
 Intermediate Storage • 5  
 Interrupt • 8  
 Interval • 7  
 Introduction • 1  
 Inverse Format Registers -- modbus • 142

## L

Lead Length • 46  
 Leaf Node • 133, 134  
 Lightning • 55, 56, 4  
 Lightning Protection • 56  
 Lightning Rod • 56  
 Link Performance • 137  
 LoggerNet • 144, 145  
 Logical Expression • 88  
 Logical Operators • 100  
 Loop Counter • 5  
 Low-Level ac • 45

## M

Maintenance • 32, 147  
 Manual Organization • 1  
 Manually Initiated • 5  
 Math • 87, 100, 153  
 Mathematical Operation • 87  
 Measurement -- Accuracy • 46, 55, 1, 13  
 Measurement -- Instruction • 84, 98  
 Measurement -- Peripheral • 49  
 Memory • 88, 129  
 Memory Conservation • 88, 130  
 Memory Reset • 130  
 Milli • 5  
 Modbus • 107, 139, 141, 5  
 Moisture • 32  
 Monitoring Data • 11, 18  
 Mounting • 4  
 Multi-Meter • 6  
 mV • 6

## N

NAN • 153, 6  
 Neighbor Device • 6  
 Network • 134  
 Network Planner • 145  
 NIST • 6  
 Node • 6  
 Nodes • 133  
 Noise • 40  
 Nominal Power • 29  
 Not-A-Number • 153  
 Null-Modem • 3, 6  
 Numerical Format • 71

## O

- Offset • 86
- Ohm • 6
- Ohms Law • 7
- On-Line Data Transfer • 7
- Operating System • 60
- Operator • 100
- OS • 60
- OS Date • 16
- OS Version • 16
- Output • 7
- Output Array • 7
- Overview • 23
- Overview -- Modbus • 139
- Overview -- Power Supply • 53

## P

- PakBus • 105, 133, 7, 33
- PakBus Information • 31, 7
- PakBus Overview • 133
- Parameter • 7
- Parameter Type • 85
- PC Program • 13, 69
- PC Support Software • 34, 143
- PC200W • 11, 143
- PC400 • 144
- PDA Support • 145
- Peer-To-Peer • 105
- Period Average • 8
- Peripheral • 8
- Peripherals • 29
- Pin Out • 21
- Ping • 137
- PLC • 139, 140, 5
- Polar Sensor • 123
- Polarity Reversal • 53
- Port • 8
- Power • 155, 12
- Power Consumption • 53
- Powering Sensor • 37
- Precision • 37, 8, 13
- Predefined Constant • 76
- Primer • 3
- Print Device • 8
- Print Peripheral • 8
- Priority • 83
- Processing • 100
- Processing -- Output • 94
- Processing -- Spatial • 103
- Processing -- Wind Vector • 120
- Processing Instructions • 8
- Processing Instructions -- Output • 81

- Program • 13, 93
  - Program -- Alias • 77, 93
  - Program -- Array • 103
  - Program -- Constant • 76
  - Program -- Data Storage Processing instruction • 81
  - Program -- Data Table • 77
  - Program -- Data Type • 75, 77
  - Program -- DataInterval () • 81
  - Program -- Declaration • 77
  - Program -- Dimension • 93
  - Program -- Documenting • 69
  - Program -- Execution • 83
  - Program -- Expression • 86, 153
  - Program -- Floating Poing Arithmetic • 75, 87, 153
  - Program -- Instruction • 84
  - Program -- Mathematical Operation • 87
  - Program -- Measurement Instruction • 98
  - Program -- Modbus • 139
  - Program -- Multiplier • 86
  - Program -- Offset • 86, 120
  - Program -- Output Processing • 81, 94
  - Program -- Parameter Type • 85
  - Program -- Resource Library • 109
  - Program -- Scan • 72, 83
  - Program -- Structure • 72
  - Program -- Subroutine • 83
  - Program -- Task Priority • 83
  - Program -- Timing • 83
  - Program -- Unit • 93
  - Program -- Variable • 93, 11
- Program DataTable () • 77
- Program Editor • 69
- Programming • 13, 69, 143
- Protection • 27, 32, 147
- Protocols Supported • 132
- Pulse • 7, 8
- Pulse Count • 42
- Pulse Input • 7
- Pulse Input Channels • 42
- Pulse Measurement • 42
- Pulse Sensor • 43

## Q

- Quickstart Tutorial • 3

## R

- Range Limit • 37
- Record Number • 106
- Recorder • 99, 117, 118
- Regulator • 155, 8
- Relay • 50
- Relays • 50

Reliable Power • 53  
 Requirement -- Power • 53  
 Reset • 60  
 Resistance • 3, 6, 7, 9  
 Resistive Bridge • 6, 41  
 Resistor • 9  
 Resolution • 9, 13  
     Resolution -- Concept • 13  
     Resolution -- Data Type • 75  
     Resolution -- Definition • 9  
 Retrieving Data • 131, 132  
 Ring Memory • 4, 9  
 RMS • 9  
 Router • 31  
 RS-232 • 154, 9  
 RS-232 Pin Out • 21  
 RS-232 Port • 9, 21  
 RTDAQ • 144, 3  
 RTU • 140  
 Runtime Signature • 10

## S

Sample Rate • 9  
 Satellite • 108  
 SCADA • 107  
 Scan • 9  
 Scan (Execution Interval) • 9  
 Scan Interval • 72, 9  
 Scientific Notation • 71  
 SDI-12 • 112, 10  
     SDI-12 Measurement • 153  
     SDI-12 Recording • 46  
     SDI-12 Sensor • 47, 112  
 Self-Calibration • 33, 41  
 Send • 10  
 Sensor • 3  
     pulse • 7, 42, 8  
     Sensor -- Analog • 6  
     Sensor -- Bridge • 6  
     Sensor -- Frequency • 8, 43  
     Sensor -- Period Average • 8  
     Sensor -- Resistive Bridge • 41  
     Sensor -- Serial • 28, 47  
     Sensor -- Sine Wave • 25  
     Sensor -- Square Wave • 25, 43  
     Sensor -- Voltage • 3, 6, 153  
 Sensor Support • 37, 112  
 Sequence • 38  
 Serial • 10  
 Serial I/O • 105  
 Serial Input • 105  
 Serial Sensor • 47

Setting • 12, 30  
     Setting -- CRBASIC • 66  
     Setting -- Durable • 67  
     Setting -- Terminal Emulator • 66  
     Setting- PakBus • 18, 33  
 Settings Editor • 133  
 Short Cut • 13, 69  
 SI -- Systeme Internationale • 10  
 Signed Packet • 31  
 Sine Wave • 7  
 Single-Ended • 45, 57  
 Skipped Scan • 151  
 Software • 11, 34, 143, 11  
 Software -- Beginner • 143  
 Solar Panel • 158  
 Spark Gap • 55  
 Spatial Processing • 103  
 Specifications • 37  
 Square Wave • 44  
 SRAM • 30  
 Standard Deviation • 90, 94, 103  
 Starter Software • 34, 143  
 State • 11  
 Status • 106  
 status table • 15  
 Storage • 129  
 Strain • 41  
 Structure -- Program • 72  
 Subroutine • 83  
 Supply • 155  
 Support Software • 143, 3  
 SW Battery • 28  
 SW-12 • 50  
 Synchronous • 11  
 System Clock • 66  
 Systeme Internationale • 10

## T

Table • 72, 77  
 Table -- Data Header • 77  
 Telecommunication • 131  
 Temperature Range • 37, 147  
 Terminal Emulator • 59, 66  
 Terminal Emulator Settings • 66  
 Terminal Input Module • 51  
 Thermistor • 41, 98  
 Timestamp • 79  
 TIMs • 51  
 TLL • 11  
 Tutorial • 3  
 Tutorial Exercise • 10

## **U**

UPS • 11  
User Program • 30  
UTC Offset • 66

## **V**

Vac • 12  
Variable • 70, 73  
Variable Array • 74  
Variable Declaration • 73  
Variable Modifier • 76  
Variable Out of Bounds • 151  
Vdc • 12  
Vector • 120  
Verify Interval • 135, 136  
Viewing Data • 20  
Visual Weather • 143  
Volt Meter • 12

Voltage Measurement • 27, 38  
Volts • 12

## **W**

Watchdog Error • 151, 16  
Watchdog Timer • 12  
Weather Tight • 12  
Wind Vector • 120  
Wind Vector Processing • 120  
Wiring • 15  
Wiring Panel • 4  
Writing Program • 13, 69

## **X**

XOR • 88, 100

## **Z**

Zero • 88





## Campbell Scientific Companies

---

**Campbell Scientific, Inc. (CSI)**

815 West 1800 North  
Logan, Utah 84321  
UNITED STATES

[www.campbellsci.com](http://www.campbellsci.com) • [info@campbellsci.com](mailto:info@campbellsci.com)

**Campbell Scientific Centro Caribe S.A. (CSCC)**

300 N Cementerio, Edificio Breller  
Santo Domingo, Heredia 40305  
COSTA RICA

[www.campbellsci.cc](http://www.campbellsci.cc) • [info@campbellsci.cc](mailto:info@campbellsci.cc)

**Campbell Scientific Africa Pty. Ltd. (CSAf)**

PO Box 2450  
Somerset West 7129  
SOUTH AFRICA

[www.csafrica.co.za](http://www.csafrica.co.za) • [cleroux@csafrica.co.za](mailto:cleroux@csafrica.co.za)

**Campbell Scientific Ltd. (CSL)**

Campbell Park  
80 Hathern Road  
Shepshed, Loughborough LE12 9GX  
UNITED KINGDOM

[www.campbellsci.co.uk](http://www.campbellsci.co.uk) • [sales@campbellsci.co.uk](mailto:sales@campbellsci.co.uk)

**Campbell Scientific Australia Pty. Ltd. (CSA)**

PO Box 8108  
Garbutt Post Shop QLD 4814  
AUSTRALIA

[www.campbellsci.com.au](http://www.campbellsci.com.au) • [info@campbellsci.com.au](mailto:info@campbellsci.com.au)

**Campbell Scientific Ltd. (CSL France)**

3 Avenue de la Division Leclerc  
92160 ANTONY  
FRANCE

[www.campbellsci.fr](http://www.campbellsci.fr) • [info@campbellsci.fr](mailto:info@campbellsci.fr)

**Campbell Scientific (Beijing) Co., Ltd.**

8B16, Floor 8 Tower B, Hanwei Plaza  
7 Guanghua Road  
Chaoyang, Beijing 100004  
P.R. CHINA

[www.campbellsci.com](http://www.campbellsci.com) • [info@campbellsci.com.cn](mailto:info@campbellsci.com.cn)

**Campbell Scientific Ltd. (CSL Germany)**

Fahrenheitstraße 13  
28359 Bremen  
GERMANY

[www.campbellsci.de](http://www.campbellsci.de) • [info@campbellsci.de](mailto:info@campbellsci.de)

**Campbell Scientific do Brasil Ltda. (CSB)**

Rua Apinagés, n.br. 2018 — Perdizes  
CEP: 01258-00 — São Paulo — SP  
BRASIL

[www.campbellsci.com.br](http://www.campbellsci.com.br) • [vendas@campbellsci.com.br](mailto:vendas@campbellsci.com.br)

**Campbell Scientific Spain, S. L. (CSL Spain)**

Avda. Pompeu Fabra 7-9, local 1  
08024 Barcelona  
SPAIN

[www.campbellsci.es](http://www.campbellsci.es) • [info@campbellsci.es](mailto:info@campbellsci.es)

**Campbell Scientific Canada Corp. (CSC)**

14532 – 131 Avenue NW  
Edmonton AB T5L 4X4  
CANADA

[www.campbellsci.ca](http://www.campbellsci.ca) • [dataloggers@campbellsci.ca](mailto:dataloggers@campbellsci.ca)

Please visit [www.campbellsci.com](http://www.campbellsci.com) to obtain contact information for your local US or international representative.